PREVENTING ABUSE OF ONLINE COMMUNITIES

A Thesis
Presented to
The Academic Faculty

by

Danesh Irani

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Computer Science

Georgia Institute of Technology
August 2012

PREVENTING ABUSE OF ONLINE COMMUNITIES

Approved by:

Professor and John P. Imlay Jr. Chair in Software, Dr. Calton Pu, Advisor
School of Computer Science
Georgia Institute of Technology

Associate Professor, Dr. Kang Li
School of Computer Science
Georgia Institute of Technology

Professor, Dr. Mustaque Ahamad
School of Computer Science
Georgia Institute of Technology

Assistant Professor, Dr. Jonathon Giffin
School of Computer Science
Georgia Institute of Technology

Professor, Dr. Ling Liu
School of Computer Science
Georgia Institute of Technology

Date Approved: May 8^{th}, 2012

To my family and all those who have supported me through this journey.

ACKNOWLEDGEMENTS

"It takes a village to raise a child" is certainly true in the journey to obtain a Ph.D. My journey has been a transformation not only in terms of being able to conduct and publish research, but also in terms of personal growth. I am truly grateful and sincerely thankful to everyone who has been there through the peaks and valleys of this journey.

First and foremost I want to thank my advisor, Prof. Calton Pu. His constant guidance, inspiration, and encouragement have been one of the driving reasons for my transformation during the Ph.D. He pushed me to exceed my own boundaries and excel in all aspects of what I did, without taking any shortcuts. I cannot understate my appreciation towards the freedom he gave me to pursue my interests within the area of security and privacy, as well as for his patience during my exploration of the area. I admit that initially I did not truly understand the value of his meta-comments but I have realized that over-time they have been invaluable to making me a better and more capable researcher.

I would also like to thank members of my dissertation committee – Prof. Mustaque Ahamad, Prof. Jonathon Giffin, Prof. Ling Liu, and Prof. Kang Li – for reading and commenting on my dissertation. My proposal and defense were two of the most grueling and fun presentations I have done, and I appreciate all the questions and challenges that led to an improved dissertation.

To Prof. Jonathon Giffin and Prof. Kang Li, co-authoring papers with you was a pleasure and I appreciate all the careful critiques of my work and the advice given to me both in relation to research and life. I am also thankful to my other co-authors including Prof. Davide Balzarotti, Prof. Engin Kirda, and Prof. Lakshmi Ramaswamy whom I have all learned a great deal from. I would like to especially thank Prof. Engin Kirda and Prof. Davide Balzarotti for believing in me and advising me during my time as a visiting researcher at Eurecom, and after.

To all my internship mentors including Darren Shou, Sanjay Sawhney, Dr. Susanta

Nanda, Diego Gilscarbo, Brian Williams, Dr. Nitya Narasimhan, and Tzvetan Horozov, I thank you for all your guidance and everything I have learned from you. It was amazing having such great people to work with and I appreciate it dearly.

To Mehenaz Soonawalla, Christina Lee, and Monika Chhabria, more affectionately known as the council of wise women. You have supported me all along and have been there to hear about the good times and the bad, in as much or as little detail as I wanted to give. You have always known the right thing to say or sometimes just to listen. For all this and more, my sincerest thanks.

To Dr. Steve Webb, your drive and dedication inspire me; and I will not forget your guidance and patience with a junior Ph.D. student entering the area of security and privacy. I thank you for the early direction in research and for all the games of disc golf. To Dr. Qinyi Wu – a caring labmate who constantly reminded me to take better care of myself and did not fail to ask me the hard questions others may have sidestepped –, to Dr. Bhuvan Bamba – for answering questions tirelessly and making great food –, and to Dr. Vishakha Gupta-Cledat – who spent many hours studying together for the System qualifier exam – thank you.

To friends who have peppered the journey to my dissertation with cherished memories, including Ahamad Al-Yamani, Vijay Balasubramiam, Leyla Bilge, Martim Carbone, Rachita Chum, Romain Cledat, Nevena Djaja, Ioannis Doudalis, Kyle Gochenour, Vishaka Gupta, Binh Han, Andrea Lanzi, Athena Lee, Sarah Iqbal, Yani Ioannou, Sarishta Katrak, Vahishta Katrak, Mukil Kesavan, Tushar Kumar, Zohre Kurt, Sarah Naqvi, Devi Sampat, Mona Shah-Joshi, Kapil Singh, Nadia Szeinbaum, Bianca Su, Priyanka Tembey, De Wang, Qingyang Wang, and Andrea Zohner my sincerest thank you!! To any labmates and friends at Georgia Tech, the hours in the lab would have gone by a lot slower without your company and I would have enjoyed it a lot less – thank you. To my labmates and friends at Eurecom, my time in Sophia Antipolis was undoubtedly one of the best summers of my life – thank you. To friends in Toronto, coming home in winter would not be as warming without you.

Last, but not least, to my family – words cannot express my gratitude for always being there and encouraging me along the way. I am all I am because of you and I love you all

dearly. To my father, Sharukh Irani, I hope to someday be as good a father as you. To my brother, Shahvir Irani, thanks for being the best brother anyone could ask for. To my mother, even though you are not here, this is for you.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Online communities are growing at a phenomenal rate and with the large number of users these communities contain, attackers are drawn to exploit these users. Denial of information (DoI) attacks and information leakage attacks are two popular attacks that target users on online communities. These information based attacks are linked by their opposing views on low-quality information. On the one hand denial of information attacks which primarily use low-quality information (such as spam and phishing) are a nuisance for information consumers. On the other hand information leakage attacks, which use inadvertently leaked information, are less effective when low-quality information is used, and thus leakage of low-quality information is preferred by private information producers.

In this dissertation, I introduce techniques for preventing abuse against these attacks in online communities using meta-model classification and information unification approaches, respectively. The meta-model classification approach involves classifying the "connected payload" associated with the information and using the classification result for the determination. This approach allows for detection of DoI attacks in emerging domains where the amount of information may be constrained. My information unification approach allows for modeling and mitigating information leakage attacks. Unifying information across domains followed by a quantification of the information leaked, provides one of the first studies on users' susceptibility to information leakage attacks. Further, the modeling introduced allows me to quantify the reduced threat of information leakage attacks after applying information cloaking.

CHAPTER I

INTRODUCTION

With the growth of the Internet and increase in the time spent online, online communities are growing at a phenomenal rate. These online communities often exist via information systems which range from early online communities such as bulletin board systems (BBS) and email systems to modern online communities such as social networks. Members of the online community usually share common interests or know each other in real life, and participate in the online community by exchanging information from messages to pictures and music.

Some of the most popular online communities are social networks, with large social networks such as Facebook, Twitter, and MySpace attracting millions of visitors [40]. These online communities allow users to communicate with friends and strangers alike, usually first requiring the creation of an online profile. A user, using his/her online profile, can easily share information such as personal pictures, life events, thoughts, and music to other online profiles (users).

A large number and variety of online communities have emerged primarily due to each online community catering to a facet of a user's life [34]. For example, Facebook for personal networking, Flickr for picture sharing, LinkedIn for professional networking, and Twitter for keeping updated with events. This mirrors real-life where a person may behave differently with different sets of people and even have different sets of friends for some areas of his/her life. For example, a person is a musician to one set of individuals and a teacher to another set of individuals. Similarly in online communities, users sign-up for multiple online profiles [65] and share different facets of their lives with different sets of users.

With the large number of users and information contained within these online communities, attackers are attracted to try and abuse them. Although there are many different ways to abuse social networks, the two we focus on are diametrically opposing views of

low-quality information. On the one hand, from an *information consumer's standpoint*, low-quality information is undesired as it detracts from consumption of useful information in the online community. In extreme cases, namely denial of information (DoI) attacks this can make communities unusable, especially in cases of floods of spam, phishing, or fake information. On the other hand, from a *private information producer's standpoint*, if his information is leaked to attackers, their obtaining low-quality information is desired as it reduces the effectiveness of potential attacks. Users are currently unaware of the dangers of unintended information leakage, which can be used in attacks such as compromising accounts (via answering password recovery questions), stalking, and personal identification.

In this dissertation, we introduce techniques for protecting against denial of information attacks and information leakage attacks in online communities using meta-model classification and information unification approaches, respectively. Our meta-model classification approach involves using the "connected payload" associated with information in determining the label for the information begin classified. This approach allows for early-detection of DoI attacks in emerging domains where the amount of information may be constrained. As an example, a DoI attack using tweets on Twitter is difficult to detect using traditional machine learning techniques due to limited information available within 140 characters. We use the tweet content and fetched webpage meta-models to detect such attacks. We look at static profile content classification in an effort to build a meta-model classifier for social profiles. We explore the limits of static profile content classification as a stand-alone technique for early detection of social spam. We apply insights gained from analysis of the evolution of low-quality information to measure the resilience of our proposed approaches to adaptations by attackers. Evolution and adaption to new techniques is a result of the high-stakes involved in spreading low-quality information and is akin to an arms-race between spammers and spam-researchers.

Our information unification approach combines information across domains to measure personal information leakage and susceptibility to information leakage attacks. As a first step in doing so, we introduce a method using pseudonyms to build a user's social footprint,

or a combined footprint of a user's online presence. As an example, we measure the effectiveness of knowing one pseudonym is in guessing pseudonyms used on other communities and the effectiveness in guessing pseudonyms using a user's name. Using social footprints, we then introduce four measures to quantify the amount of information revealed and susceptibility of users to information leakage attacks. Having four different measures allows fine-grained determination the amount of information leaked including consistency. We use information cloaking as a measure to reduce the amount of inadvertent public information leakage.

1.1 Dissertation Statement and Dissertation Contributions

Before proceeding to the concrete contributions of this dissertation, the dissertation statement can be formulated as follows:

> Meta-model classification and information unification techniques can be used to more effectively address significant challenges in preventing abuse of online communities.

To support the dissertation statement, we make the following concrete contributions divided into two parts based on the viewpoints of low-quality information.

Part I: Protection against denial of information (DoI) attacks – Low-quality information from an information consumer's standpoint

- Our first contribution is an analysis of the evolution of low-quality information which occurs as a result of an arms-race between spammers and spam-researchers. Specifically, we investigate characteristics and features of phishing messages – a particularly nefarious type of low-quality information disguised as important legitimate information. One of the key outcomes is identifying transitory and pervasive features associated with flash and non-flash attacks. We find that transitory features are usually strong indicators of phishing which is likely the reason that the features appear only for a short duration. We apply these insights in our subsequent research to develop resilient long-term classification methods.

- Our second contribution explores the effectiveness and long-term suitability of profile classification using only static content. Static content is available on profile creation – the entry point to sending spam on a social network – and thus we propose classification on this content to identify and preclude spam profiles before spammers can use them to propagate spam in the online community. Using insight from our previous contribution, we evaluated features along the discriminatory power and robustness dimensions, identifying features most likely to be transitory. We then measured classification performance with all the features as well as after removal of in transitory features. We found that our classifiers performed well in both instances, but classification using text-features was more robust to removal of transitory features than classification using categorical features.

- Our third contribution investigates identification of low-quality messages sent in online communities where content rich profiles are not available. With message lengths limited to a few hundred characters in some online communities, traditional machine-learning techniques which are used on content rich messages will be ineffective (e.g. blog posts or personal messages). Although our previous contribution would be effective given static profile content, in the case of Twitter the message content is limited to 140 characters as is the profile content to 160 characters. We introduce a meta-model classification technique which combines classification of the short tweet-text along with additional fetched content (such as linked web pages) to improve our accuracy. We find that fetching associated content improves our classifiers F1-measure by over 13.7%.

Part II: Protecting against information leakage attacks – Low-quality information from a private information producer's standpoint

Our approach involves unifying profiles across social networks to measure the amount of information leaked and susceptibility towards information leakage attacks. After introducing information cloaking, we quantify the reduction in users' susceptibility towards information leakage attacks.

- Our fourth contribution proposes a technique to unifying a person's profile (i.e., finding social profiles associated with a single user) across online communities utilizing pseudonyms. We find that over 40% of a person's online identity can be reconstructed by using a single pseudonym, and by using the person's name the attacker can reconstruct 10% to 35% of a person's online identity.

- Our fifth contribution introduces models to measure the amount of public information released across unified user's profiles. We find, based on over 8,200 user's online identities, that the number of users susceptible to two real-world attacks increases approximately nine times when looking at users' unified social profile in comparison to users' individual social profile. We propose four measures to quantify information revealed by a user's unified social profile. The first measure quantifies the amount of information revealed by the unified social profile; the second, quantifies the amount of information revealed and consistent in a unified social profile using Entropy from Information Theory; the third, quantifies the amount of information revealed and consistent in a unified social profile using Guessing Entropy; and the last, quantifies the amount of consistent information revealed in a unified social profile. We use these measures to build a framework for detecting the susceptibility of a user towards an information attack, such as a personal identification attack or a password recovery attack. Using the same dataset of 8,200 user's online identities, we find that the number of users susceptible to attacks increases approximately nine times when looking at a user's combined social footprint. We then use information cloaking to reduce the usefulness of information publicly released by reducing its granularity. For example, instead of publicly releasing an age of 25, a social network might indicate that the user is between the ages of 20-25.

- Our final contribution is demonstrate social engineering attacks against online communities. We show that although most people hide information by making details of their profile private, we can use reverse social engineering to engage users. That is, we discuss and show how attackers in practice can abuse some of the friend-finding

features that online social networks provide with the aim of making users contact them. Our results demonstrate that reverse social engineering attacks are feasible and effective in practice.

1.2 Organization of the Dissertation

We split each step of each contribution into a separate chapter, with chapters separated by parts, to emphasize the different viewpoints of low-quality information being tackled. We attempt to keep each chapter an independent unit with its own evaluation and related work. Common techniques and algorithms used across chapters are summarized in the chapter "Preliminaries". The remained of the dissertation is organized as follows:

- **Chapter II: Preliminaries** – We review common techniques used in this dissertation. machine learning techniques used in this dissertation. In addition, we some attribute selection methods.

Part I: Protection against denial of information (DoI) attacks

Low-quality information from an information consumer's standpoint

- **Chapter III: Evolution of Phishing** – To begin to understand the adversarial evolution of low-quality information, we study the evolution of phishing email messages using a corpus consisting of over a year of phishing messages. We identify phishing campaigns within this corpora and characterize phishing campaigns as *flash-attacks* or *non-flash attacks*. In addition we identify *transitory* features and *pervasive* features present in these attacks. We find features which are present in a few attacks and have a relatively short life span (transitory) are generally strong indicators of phishing, whereas features which are present in most of the attacks and have a long life span (pervasive) are generally weak selectors of phishing.

- **Chapter IV: Detection of Social Spam Profiles using Static Content** – We make a case for analysis of static user profile content, possibly as soon as such profiles are created, through an experimental study of over 1.9 million MySpace profiles. We

compare several machine learning algorithms in their ability to distinguish spam profiles from legitimate profiles. We found that a C4.5 decision tree algorithm achieves the highest accuracy (99.4%) of finding rogue profiles, while naïve Bayes achieves a lower accuracy (92.6%). Having studied adversarial behavior, we also conducted a sensitivity analysis of the algorithms w.r.t. features which may be easily adapted/removed by spammers.

- **Chapter V: Detection of Trend-stuffing on Twitter** – In cases where a lot of static user profile content is not available and only a small amount of message text is available, we introduce meta-model classification to improve accuracy. We study the use of text-classification over 600 trends consisting of 1.3 million tweets and their associated webpages to identify tweets closely-related to a trend and in doing so finding unrelated tweets. We compare the use of naïve Bayes, C4.5 decision trees, and Decision Stumps, over the individual sets of features, followed by combining predictions of classifiers over the tweet text and associated web-page text. In addition, we look at the resilience of these classifiers to adversarial behavior by emulating an adversary and rerunning our experiments.

Part II: Protecting against information leakage attacks

Low-quality information from a private information producer's standpoint

- **Chapter VI: User's Social Footprints** – We study large online social footprints by collecting data on 13,990 active users. We investigate the ease by which an attacker can reconstruct a person's social network profile. Namely, we find that over 40% of an individual's social footprint can be reconstructed by using a single pseudonym (assuming the attacker guesses the most popular pseudonym), and an attacker can reconstruct 10% to 35% of an individual's social footprint by using the person's name. We also perform an initial investigation of matching profiles using public information in a person's profile.

- **Chapter VII: Modeling Information Leakage** – We model a user's social footprint and introduce four measures to quantify the amount of information revealed in respect to information leakage attacks (such as, answering password reset questions to compromise user accounts and stalking). Using the online social footprints collected in the previous chapter, we measure user's susceptibility towards the attacks. Finally, we apply information cloaking to reduce the amount of information leakage.

- **Chapter VIII: Reverse Social-Engineering** – In most online communities, friends of a user are given a higher level of trust and access to personal information on the user. We study reverse social engineering attacks, or social engineering attacks in which the target user interacts with the attacker given certain indirect stimuli. That is, we discuss and show how attackers, in practice, can abuse some of the friend-finding features that online social networks provide with the aim of launching reverse social engineering attacks and friending other users. Finally, we discuss mitigation methods to secure information on online communities.

- **Chapter IX: Conclusion** – In this chapter, we wrap up the dissertation.

CHAPTER II

PRELIMINARIES

In this chapter we cover some of the fundamental techniques commonly used through this dissertation. Topics include machine learning and dimensionality reduction.

2.1 Supervised learning

Supervised learning [58] is a branch of machine learning which uses labeled data to infer a function to label future data. An input instance, usually a set of features, are provided to a learning algorithm with a set of corresponding labels (such as "Spam" and "Not-spam"). The supervised learning algorithm takes these inputs and infers a function which provides labels to most of the input data. In most cases, we use learning algorithms which infer functions that provide discrete labels (i.e. on a set of possible values), and these are known as classification algorithms or classifiers. Functions inferred by these classifiers are often referred to as classification models (or models). The input to these functions are referred to as attributes or instances, with each set of attributes or features corresponding to a particular data instance (akin to a data point), and the output representing a class label.

In evaluating supervised learning algorithms, classification is generally performed in two stages: 1) training or building a model on part of the labeled data; 2) testing or measuring performance of the classifier on a withheld part of labeled data. This method is called cross-validation [57] and is performed by creating random partitions of the labeled data into test and training sets. Unless otherwise specified, we perform 10-fold cross-validation in our experiments, where a single fold is one run of the cross-validation technique.

Cross-validation is also useful in preventing over-fitting – over-fitting occurs when a classifier learns the training data extremely well, such that it performs poorly on test data. We omit the details of using cross-validation for this purpose as it is out of scope of this dissertation. Stone provides a good explanation of the technique in his work [92].

Supervised learning algorithms are commonly used in security applications, especially in

applications to determine importance, spamminess, or priority of documents and messages. A concrete example would be a spam email filter being trained on spam and legitimate emails, followed by being applied to new incoming emails providing the label "Spam" or "Not-spam" based on the previously learned model of "Spam"/"Not-spam".

2.1.1 Standard Supervised Learning Algorithms

We use a number of standard supervised learning algorithms, which we discuss below:

2.1.1.1 Naïve Bayes

Naïve Bayes [17, 71] is a probabilistic classifier which uses Bayes theorem [19] to calculate the probability of an instance belonging to a class given the set of features in the instance. One of the characteristic features of naïve Bayes is the strong assumption of independence between each pair of input attributes on the output class – namely, naïve Bayes assumes that a feature value is independent of any other feature value, given the output class.

Calculating the probability of a class given a single feature would use the standard Bayes theorem below, where C is the class and F represents the feature:

$$\mathbb{P}(C|F) := \frac{\mathbb{P}(F|C)\mathbb{P}(C)}{\mathbb{P}(F)}$$

Extending this to a set of features, the joint probability of a class label can be calculated using:

$$\mathbb{P}(C|F_1, \ldots, F_n) := \frac{\mathbb{P}(F_1,\ldots,F_n|C)\mathbb{P}(C)}{\mathbb{P}(F_1,\ldots,F_n)}$$

When calculating the *maximum a posteriori* label, the term $\mathbb{P}(F_1, \ldots, F_n)$ can be dropped as it is independent of C. Also due to the strong assumption of independence between features, we can further simplify the formula to:

$$\mathbb{P}(C|F_1, \ldots, F_n) := \mathbb{P}(F_1|C)\mathbb{P}(F_2|C)\ldots\mathbb{P}(F_n|C)\mathbb{P}(C)$$

Naïve Bayes is well suited for high dimensionality data primarily because the assumption of independence between attributes results in a low computation cost and alleviates effects of the curse of dimensionality. This has made naïve Bayes one of the standard algorithms

applied to problems requiring text-classification. Although intuitively one might expect the assumption of independence between features to result in poor performance, it has been shown in practice to evenly match and even outperform sophisticated classification methods. Further, the naïve Bayes classifier can be easily trained as a passive or active learning classifier.

2.1.1.2 C4.5 Decision Trees

A *C4.5 decision tree* is a decision tree classifier. It uses a top-down approach in building the decision tree. At the root node, it picks a feature with the highest normalized Information Gain (explained in Section 2.3). Subsequent splits are made at each of the branches using features with the highest information gain on the subset of data. Complexities of this algorithm including the branch-pruning and handling of continuous features (as opposed to discrete features) are omitted. Details can be found in Quinlan's book [82].

Determining labels for new instances of data is fairly quick with decision trees – the primary reason being that determining a class label involves a traversal of the tree to a leaf. Further, this algorithm is commonly used due to the decision trees being easily readable and providing insight when analyzing results.

2.1.1.3 Support Vector Machine (SVM)

A *Support Vector Machine (SVM)* classifier creates a decision boundary in hyperspace which achieves maximum separation of the data points. In this case, each data instance is considered an N-dimensional data point, where every dimension corresponds to a feature. A simple SVM providing a hyperplane separating the data points can be calculated by solving the following optimization problem:

$$\min_{w,\xi,b} \left\{ \frac{1}{2}\|w\|^2 - C\sum_{i=1}^{n} \xi_i \right\}$$

$$\text{subject to (for any } i = 1, \ldots, n)$$

$$y_i(w \cdot x_i - b) \geq 1 - \xi_i,\ \xi_i \geq 0$$

where w is the normal vector to the hyperplane, x_i is a particular data point, C a constant, b is the offset, and ξ_i is a slack variable – slack variables allow for misclassifications, but still maximizing the distances to the nearest cleanly split examples.

Given that there are often sets of data points which are not linearly separable a common technique used is applying a function – called a Kernel Function due to properties of specific functions usable for this purpose – to map the data points to a higher-dimension. This ideally will make the points easily separable. Unless mentioned otherwise, we use a polynomial kernel function for high-dimensional data and a radial basis function (RBF) kernel function otherwise.

2.1.1.4 Boosting

A *Boosting* classifier is a meta-classifier that uses a set of weak learning algorithms to create a strong learning algorithm. Weak learning algorithms are simply classifiers which provide better than random accuracy when classifying data. Typical weak learning algorithms include a decision stump or a single node neural network.

We use a popular variant of boosting, namely AdaBoost (Adaptive Boosting), which increases the importance of misclassified instances on every iteration of the boosting algorithm. Thus, at every successive iteration of the boosting algorithm focuses on classifying misclassified instances and the overall meta-classifier increases in accuracy. We use AdaBoost with Decision Stumps as the results of the classification with them are easily readable and provides insight when analyzing results. Generally only weak learners are used with boosting, otherwise the meta-model can become increasingly complex and over-fitting can occur.

2.1.1.5 Decision Stump

A *Decision Stump* is a single-level decision tree. Similar to decision trees, a feature is picked with the highest normalized Information Gain (explained in Section 2.3) for the single internal node. Branches from this internal node lead to leaves nodes (or class labels).

Due to the simplicity of this classifier, we use it to see how easily a class can be partitioned using a single feature. Also as Decision Stumps are weak learners, they are used as

Table 1: List of classifiers used with categorical features.

Algorithm Type	***Classifier***
AdaBoost	*AdaBoostM1 (w/ DecisionStump)*
Artificial Neural Network	*Multilayer Perceptron (MLP)*
C4.5 Decision Tree	*J48*
Decision Stump	*DecisionStump*
Naïve Bayes	*Naive Bayes*
Support Vector Machine	*SMO (w/ PolyKernel)*

components of other learning algorithms, such as Boosting algorithms.

2.1.2 Implementation of Supervised Learning Algorithms

We use the standard implementation of supervised machine learning algorithms from Weka 3.6.1 [103]. Table 1 lists the standard classifier names as well as the name of the standard algorithm as implemented by Weka.

2.1.3 Evaluation Criteria

We compare classifier performance using the false-positive (FP) errors, false-negative (FN) errors, area under the curve (AUC) and F1-measure as our evaluation metrics. To define these metrics, we first review the set of possible outcomes of a single classification using a confusion matrix:

Table 2: Confusion matrix

Predicted Label	Actual Label	
	Class A	*Class B*
Class A	True-Positive (TP)	False-Positive (FP)
Class B	False-Negative (FN)	True-Negative (TN)

Assuming our labels are spam and non-spam, the false-positive and false-negative represent the number of spam instances classified as non-spam and the number of non-spam instances misclassified as spam, respectively.

The AUC can be intuitively thought of as a measure of the detection trade-off between FPs and TPs. It is calculated based upon the received operator characteristic (ROC) curve, which measures the trade-off between FP-rate and TP-rate. Using a classifier which produces a probability of an instance belonging to a particular classes, the trade-off can

simply be adjusted by varying a decision threshold, adjusting one's tolerance towards higher FPs and TPs or vice-versa. Examples of such classifiers are neural networks and naïve Bayes. ROC curves can also be applied to discrete classifiers (output binary decisions instead of probabilities), but this usually results in a graph with a single point. Due to the skewed number of positive (non-spam) instances versus negative (spam) instances, the AUC might not always be proportional to the total number of FPs and TPs because it relies on the rate of such FPs and TPs.

The F-measure is a weighted harmonic mean between the precision and recall. Precision is a measure of the fraction of actual positive labels found by the classifier in relation to all positive labels found by the classifier, given by $\frac{\mathrm{TP}}{\mathrm{TP + FP}}$. Recall is a measure of the fraction of positive labels found by the classifier in relation to all actual positive labels, given by $\frac{\mathrm{TP}}{\mathrm{TP + FN}}$. Finally the F-measure is given by $\frac{(1+\beta 2)*TP}{\beta 2*TP+FP+FN}$. The F1-measure is the F-measure with the value of β set to 1.

2.2 Standard Unsupervised Learning Algorithms

Unsupervised learning is a branch of machine learning which tries to group similar instances within unlabeled data. More formally this can be thought of identifying structure within unlabeled data. With unlabeled data, the learning algorithms rely on density estimation. The main unsupervised algorithms used are Expectation-Maximization [32] and K-Means [68].

We do not discuss details of these algorithms here as unsupervised learning (or clustering) was mainly used in mining the data when trying to explain results or phenomena observed.

2.3 Dimensionality Reduction

Dimensionality reduction is the process of reducing the number of dimensions being considered by a classifier. We focus primarily on feature selection, which involves choosing a subset of features—typically the strongest ones. Methods for feature selection include information gain and the χ^2 test.

2.3.1 Information Gain

Information Gain, from an information theory perspective, is a measure of the amount of expected reduction in entropy if one were to use the feature (f) to partition instances belonging to a class (c), from instances not belonging in a class. In other words, it is a measure of the expected reduction in entropy if using a feature to predict the class outcome. After calculating the amount of Information Gain for each feature, we pick the strongest features.

2.3.2 χ^2 Test

The χ^2 test is another measure of how well a feature predicts a class. More precisely, it measure the lack of independence between a feature (f) and the class (c). The χ^2 test takes into account values from a two-way contingency table between f and c, which represents the number of times f and c co-occur, the number of times f occurs without c, and so-forth. Higher values of the χ^2 test indicate a stronger discriminative power.

PREVENTING ABUSE OF ONLINE COMMUNITIES

PART I

Protection against denial of information (DoI) attacks

by

Danesh Irani

CHAPTER III

EVOLUTION OF PHISHING ATTACKS

Phishing is an online form of *pretexting*, a kind of deception in which an attacker pretends to be someone else in order to obtain sensitive information from the victim. Phishing is a significant practical problem, with reported accumulated loss of $3.2 billion in 2007 [70]. Due to the immediate monetary rewards from the sensitive information stolen (e.g. user account name and password), financial institutions such as PayPal, eBay, and banks have been the primary brands affected by phishing attacks [70].

The typical communication medium phishing attacks use is email, forged to look like it is from a legitimate organization. The email usually informs the victims of a problem with their account and directs them to take remedial action by entering personal information or logging into their account at a fake website. Although phishing has already spread to other online communities, such as instant messaging, where phishing was first reported [6], and voice-over-IP (a.k.a. vishing) [97], the focus of this study is email-based phishing due to its prevalence and data availability.

It is non-trivial to distinguish phishing messages from legitimate messages, since phishing messages are constructed to resemble legitimate messages as much as possible. The defensive techniques used to identify phishing messages commonly include collaborative filtering (e.g. user reports), blacklisting (e.g. URLs collected from honeypot email accounts) [7, 12], and text classification combined with similarity testing methods (e.g. minor variations of known phishing messages). Although collaborative filtering and blacklisting can be effective after an initial positive identification, they contain a lag time during which some percentage of the phishing messages reach the victims. Due to the similarity between phishing and legitimate messages (by design), spam-filter style text classification filters have difficulties working alone and are typically used in combination with similarity tests and blacklists.

Not surprisingly, phishing message producers change the content or format of phishing

messages when they realize that previous messages have been identified or blacklisted. This constant evolution of phishing attacks results in an arms race between the generation of new phishing messages and defenses used to identify them, in an area known as adversarial classification [30].

This chapter describes an evolutionary study of phishing on a large dataset containing more than 380,000 phishing messages collected over 15 months. We aim to identify evolutionary trends in construction techniques used in phishing messages (such as construction techniques that appear and disappear). The analytical methods used are similar to a previous study on the evolution of spam messages [81]. In the study of spam messages, we were able to identify clear evolutionary trends of spam construction techniques in spam messages (namely extinction and co-existence of features) and their correlation with factors such as environmental changes and spam filtering techniques.

The main results of the study on phishing messages are a classification of phishing messages and a corresponding classification of phishing construction techniques. To this end, we identify phishing campaigns within the phishing messages by using de-duplication techniques. We find that message duplication ranges from 36.7% to 81.4% per month and given that most of these campaigns demonstrate a bursty behavior, we refer to the individual campaigns as attacks. Our first result is that phishing messages can be divided into two groups: *flash attacks* and *non-flash attacks*. Flash attacks are characterized by a large volume of phishing messages in a single campaign sent within a short period of time. Non-flash attack messages are spread over a relatively longer time span but maintain their identifiable similarity. Our second result is that the phishing message features we found can be divided into two groups: *transitory* features and *pervasive* features. Transitory features are short lived and appear in a small number of attacks, whereas pervasive features have a relatively longer lifetime and appear in a larger number of attacks.

While these classifications are very useful in helping us understand phishing messages, they also show the limitations of current techniques and tools to identify phishing messages. As a concrete example, the transitory features can be used by email filters since they are strong indicators of phishing messages, but their transitory nature shows the phishing

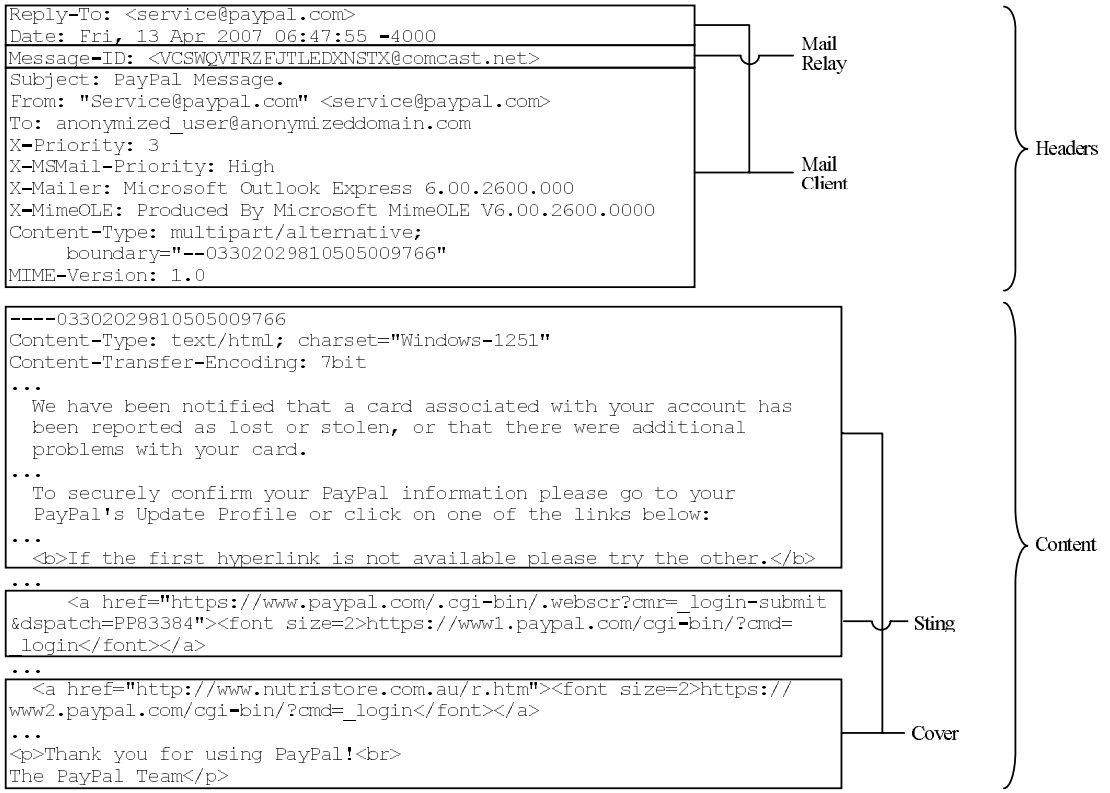


Figure 1: Example of a raw PayPal phishing message (see Figure 2 for rendered version)

message producers know this and adapt by eliminating these strong indicators in subsequent generations of phishing messages. Another example is the appearance of pervasive features in both phishing and legitimate messages, thus making them weak selectors in email filters. Consequently, our results also show the strong need for further study of phishing message identification.

3.1 Background

Phishing attacks are often filtered in a manner similar to that of spam, yet spam identification techniques applied naively to phishing messages will have a high miss rate. In this section we give an overview of the similarities and differences between phishing and spam, and then discuss the anatomy of a phishing attack.

3.1.1 Comparison to spam

The purpose of a phishing message is to acquire sensitive information about a user. In order to do so, the message needs to deceive the intended recipient into believing it is from a legitimate organization. As a form of deception, a phishing message contains no useful information for the intended recipient and thus falls under the category of spam.

Although phishing is categorized as spam, it also differs from spam. Amongst other things, spam tries to sell a product or service, while a phishing message needs to look like it is from a legitimate organization. Due to the similarity between phishing and legitimate messages, techniques that are applied to spam messages cannot be applied naively to phishing messages. For example, text-based classification [29,64,86] can perform reasonably well in identifying spam, but as a phishing message is forged to look like a message from a legitimate organization, text-based classification [35] applied naively to a phishing message will have a high miss rate.

3.1.2 Anatomy of a phishing message

A raw phishing message (Figure 1) can be split into two components: the *content* and the *headers*. These components are commonly accepted as being the major components of a message. We will use these components as a natural way to group features in the later part of the chapter.

3.1.2.1 Content

The content is the part of the message that the user sees and is used by phishing message producers to deceive users. It can be subdivided into two parts.

1. The *cover* is the content which is made to look like a message from the legitimate organization, and usually informs the user of a problem with their account. Early phishing messages could be identified based only on their cover, due to imperfect grammar or spelling mistakes (which are uncommon in legitimate messages). Over time, the covers used in phishing messages have become more sophisticated, to the point where they even warn the users about protecting their password and avoiding

Figure 2: Example of a rendered PayPal phishing message

fraud. An example of this can be seen in Figure 2, where the phishing message tells the victim to "Protect Your Account Info" by making sure "you never provide your password to fraudulent websites".

2. The *sting* is the part of the content that directs the victim to take remedial actions. It usually takes the form of a clickable URL that directs the victim to a fake website to log into their account or enter other personal details. We call this the sting, as this is the part of the content that inflicts pain, by means of financial loss or other undesirable action after the victim enters their details on the website. Typically the sting is hidden by using HTML to display a legitimate looking address, instead of the address of the fake website. An example of this is shown in Figure 1 where the address of the fake website is `http://www.nutristore.com.au/r.htm` and the corresponding displayed text in Figure 1 is a legitimate looking `https://www2.paypal.com/cgi-bin/?cmd=_login`.

3.1.2.2 Headers

The headers are the part of the message which is primarily used by the mail servers and the mail client to determine where the message is going and how to unpack the message. Most users do not see these headers, but in terms of determining if a message is phishing or not, this part of the message can be quite useful.

Headers can be subdivided into three parts based on the entities which add them to the message:

1. *Mail clients* typically add headers such as "To:", "From:", "Subject:" and some client specific headers. Examples of mail client headers are X-MSMail-Priority, X-Mailer, and X-MimeOLE, and they can be seen in Figure 1. Phishing messages may try to fake a particular header and in doing so, give away that the message is fake. For example, if the X-Mailer header indicates that a HTML message has been composed using MS Outlook but the message only contains HTML (without plain-text), this is an indication that the message is fake, as MS Outlook cannot send HTML only messages.

2. *Mail relays* will add headers along the path of the message. These are usually "Received" headers, which can be used to determine the originating IP of the message and the path taken by the message.

3. *Spam-filters or virus-scanners* will usually add headers to the message to indicate results of the tests run over the message. These headers can then be used by the receiving client to determine (based on a user-set threshold) what to do with the message.

3.2 Message Analysis

Existing email filters use a number of techniques to try and detect phishing. Some filters are based on the content of a message and use content-based collaborative filtering or text-based classification. Other filters look deeper into the construction techniques or characteristics of a message and use these as features upon which they classify.

In terms of detecting phishing using content, text-based classification (such as that used in spam) does not seem to be the best approach due to phishing messages containing text similar to that of legitimate e-mails. Content-based collaborative filtering might be a more effective content-based technique if messages have a long lifetime and a large amount of duplication. In relation to this, we measure the prevalence of duplication in phishing messages and the lifetime of duplicated messages.

Email filters which use construction techniques or characteristics of a message for classification are most effective if the feature has a long lifetime and is widely used among phishing attacks. We look at the extent to which a feature is used among phishing attacks and the lifetime of the feature. In addition, we comment on whether the feature is a strong feature (i.e. characteristic of phishing messages and does not appear in legitimate messages) or not.

3.2.1 Phishing dataset

For our analysis of phishing messages, we used messages provided by a large anti-phishing organization. The dataset contained over 1.8 million spam and phishing messages spanning 15 months from August 2006 to December 2007. The messages were submitted by users and partners of the organization as well as domain spam filtering services.

We received the dataset with each message in its own mbox, compressed and grouped by month. To avoid mis-grouped messages from adding noise to the data, we uncompressed all the messages into a single folder and sorted the messages based on their first "Received:" header. The first "Received:" header is the most reliable indication of when the message was actually delivered as it is added by the last mail relay on the path to the intended recipient and is the least susceptible to being forged. The "Date:" header can be forged and hence was not used.

3.2.2 Identifying phishing messages

The dataset we received contained both spam and phishing messages. We wanted our characterization to be limited to phishing messages, so we only used conservative techniques to identify them. We believe that this is not a limiting factor in our analysis, as the methods

we use are robust enough to be applied to larger versions of phishing corpora as well. We used three techniques to identify phishing messages.

First, we used phishing-related tests from spam-filters such as SpamAssassin. An example of a test we looked for was a SpamAssassin test called TVD_EB_PHISH. TVD_EB_PHISH checks if the message is from eBay.com and if the body of the message contains an IP based URL. IP based URLs are used by phishing message producers to host fake websites off of compromised machines that do not have DNS entries.

Second, we used headers added by domain spam filtering services. Domain spam filtering services work by taking over the Mail Exchange (MX) record of a domain, running tests on the incoming messages and then forwarding the messages to the true mail server of a domain. Usually the results of the tests run over the messages are added to the headers (e.g. X-Phishing, X-SpamSave) to indicate whether a message is phishing or not. Domain spam filtering services do not reveal the exact rules or tests that they use to determine whether a message is phishing or not, but to our knowledge (and as mentioned by one such spam filtering service [1]) they use a combination of techniques which include running SpamAssassin, URL blacklists, and checking message hashes against a database of reported phishing messages.

Lastly, we ran our own tests over the URLs in a message. Our tests check if the URL matches certain patterns which phishing message producers use to fool users into believing the URL is a legitimate address. One such pattern is if the URL contains a legitimate domain as part of a sub-domain or as part of the URL request (For example `http://www.paypal.com.phishingsite.co.uk/` or `http://www.phishingsite.com/paypal.com/`).

Using the above techniques we identified 382,377 phishing messages. The distribution of all the messages in the dataset and the distribution of only the phishing messages in the dataset are shown in Figure 3.

3.3 Phishing Content Characterization

To better understand how effective content-based collaborative filtering might be, we investigated answers to two questions:

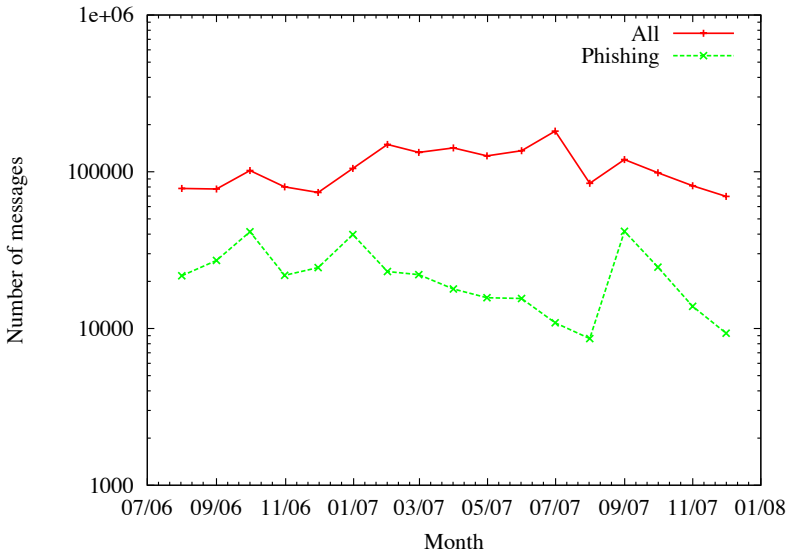


Figure 3: Distribution of phishing messages

- To what extent does duplication occur in phishing messages?
- What is the lifetime of phishing content?

As mentioned earlier, content-based collaborative filtering would be more effective if there was a large amount of duplication and if messages have a long lifetime.

3.3.1 Duplication of phishing content

To help understand the uniqueness of phishing messages, we clustered the messages in our dataset based on their rendered content (i.e., the content the user views after a email client has rendered the message's HTML content). Specifically, we used the shingling algorithm from our previous work [99] to construct equivalence classes of duplicate and near-duplicate phishing messages. Shingling provides a fuzzy approach for message comparison. In addition, it provides a storage efficient manner to represent and a computationally efficient manner to compare document similarity for a large number of documents.

The shingling algorithm we use was first discussed by Broder et al. [24], and it uses a deterministic sampling technique to map a message to a small representative set of shingles.

Messages are considered duplicates if they map onto the same set of shingles[1] and near-duplicates if they map onto similar sets of shingles. Although we use a standard version of the shingling algorithm, we discuss the basic steps for those not familiar with the technique. For a more detailed description of this shingling algorithm, please consult [36–38].

First every message's rendered content was tokenized into a collection of words, where a word is defined as an uninterrupted series of alphanumeric characters. Then, for every message, we created a fingerprint for each of its n words using a Rabin fingerprinting function [83] (with a degree 64 primitive polynomial p_A). After we had the n word fingerprints, we combined them into 5-word phrases. The collection of word fingerprints was treated like a circle (i.e. the first fingerprint follows the last fingerprint) so that every fingerprint started a phrase, which resulted in n 5-word phrases. Next, we generated n phrase fingerprints for the n 5-word phrases using another Rabin fingerprinting function (with a degree 64 primitive polynomial p_B). Once we obtained the n phrase fingerprints, we applied 84 unique Rabin fingerprinting functions (with degree 64 primitive polynomials p_1, ..., p_{84}) to each of the n phrase fingerprints, and for every one of the 84 functions, the smallest of the n fingerprints was stored. At the end of this process, each phishing message was reduced to 84 fingerprints, which are that message's *shingles*. After all of the phishing messages were converted to a collection of 84 shingles, we clustered them into equivalence classes (i.e. clusters of duplicate or near-duplicate messages).

From properties of the shingling algorithm, if two messages are 95% similar, then the probability that at least two out of the six possible non-overlapping collections match is almost 90%. If two messages are 80% similar the probability of at least two out of the six possible non-overlapping collections matching is instead only 2.6%. Two messages were considered duplicates if all of their shingles matched, and they were near-duplicates if their shingles agreed in two out of the six possible non-overlapping collections of 14 shingles.

We ran our shingling algorithm on the phishing messages, clustering duplicate and near-duplicate messages together. We were left with 137,151 unique clusters, shown in Figure 4

[1]The shingling algorithm will map a message A and message AA (message A twice) to the same set of shingles. Consequently, the definition of the term "duplicate" differs slightly from what one would expect.

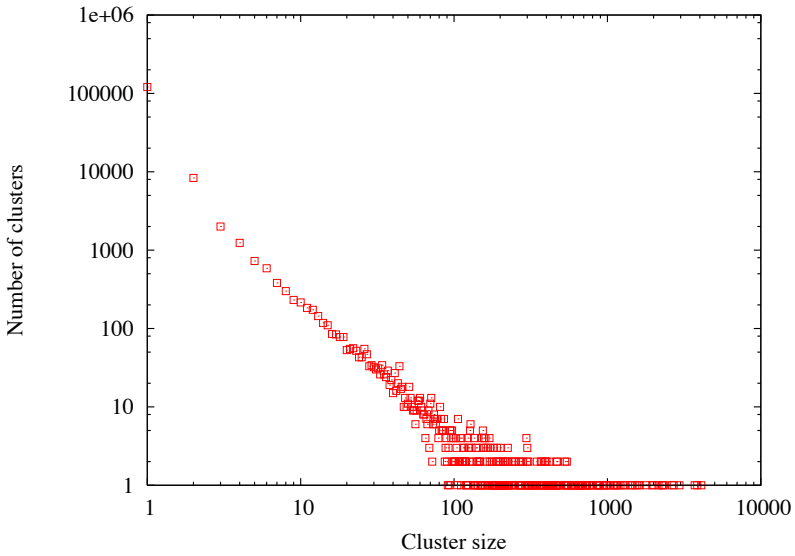


Figure 4: Distribution of Cluster Sizes (Note the Log-Log Scale)

as a distribution of of cluster sizes. From this figure, we observe that 120,439 of the clusters contain a single message. These phishing messages are truly unique because no other message in our dataset shares content with them. On the other end of the spectrum we find the largest 15 clusters containing 37,897 phishing messages, or approximately 2,526 messages per cluster. Interestingly, the remaining 16,697 clusters account for 224,041 phishing messages which means if we can identify a phishing message from the 16,697 clusters, we will be able to identify an additional 13 phishing messages on average.

Overall, only 35.9% of phishing messages are unique, and the remaining 64.1% of the messages derive their content from those unique messages. Figure 5 shows the duplication percentage by month and we see that it is significant ranging from 36.7% to 81.4% throughout the period of our data.

3.3.2 Lifetime of phishing content

To get an idea of the lifetime of a phishing content, we used the previously created clusters. We sorted them by size and picked the top 15 clusters for our analysis. For example, the top 5 clusters can be seen in Table 3.

Using the top 15 clusters, we observed that the lifetime of a cluster ranged from a month to 11 months and exhibited bursty behavior. Bursty behavior is characterized by a peak

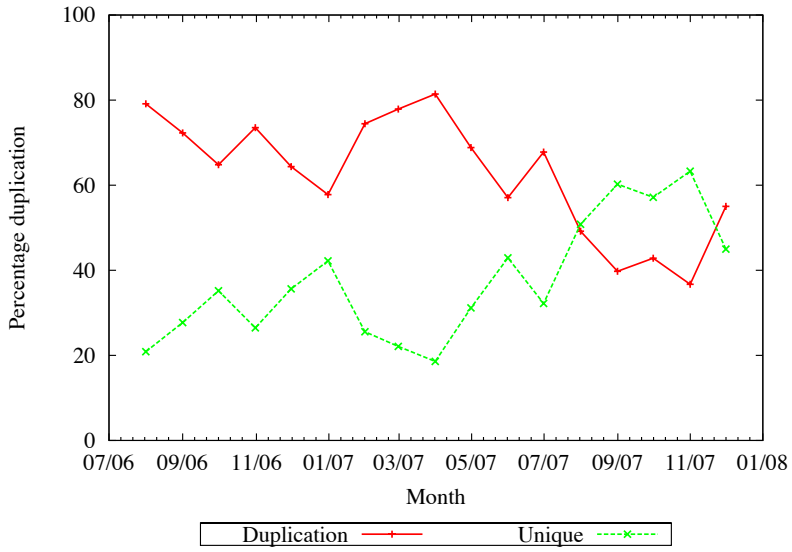


Figure 5: Phishing message duplication by month

in the number of messages sent in a month and is indicative of an attack. Based on the intensity and duration of the bursty behavior, we separated the clusters into two groups: *flash attacks* and *non-flash attacks*. *Flash attacks* are characterized by a high-intensity[2] uni-peak with a very short lifetime (under 4 months). *Non-flash attacks* are characterized by low- to medium-intensity multi-peaks and have a longer lifetime. These clusters persist from 4 to 11 months, with an average of about 8 months.

The graphed distribution of the top 15 clusters from August 2006 to December 2007 is omitted due to its complexity, but instead we show the top 15 clusters grouped by flash and non-flash attacks in Figures 6 and 7. The graphs label each cluster with its short descriptive name as well as the cluster's position in terms of size (i.e., "eBay - Malicious activity (9)" refers to a message from eBay regarding "Malicious activity" and is the 9^{th} largest cluster).

Overall, of the top 15 clusters, eight clusters were flash-attacks (shown in Figure 6), which total 22,197 phishing messages, and the other seven clusters were non-flash attacks (shown in Figure 7), which total 15,700 phishing messages.

[2]We consider a peak to be "high-intensity" if it accounts for over 4% of the phishing messages in a month.

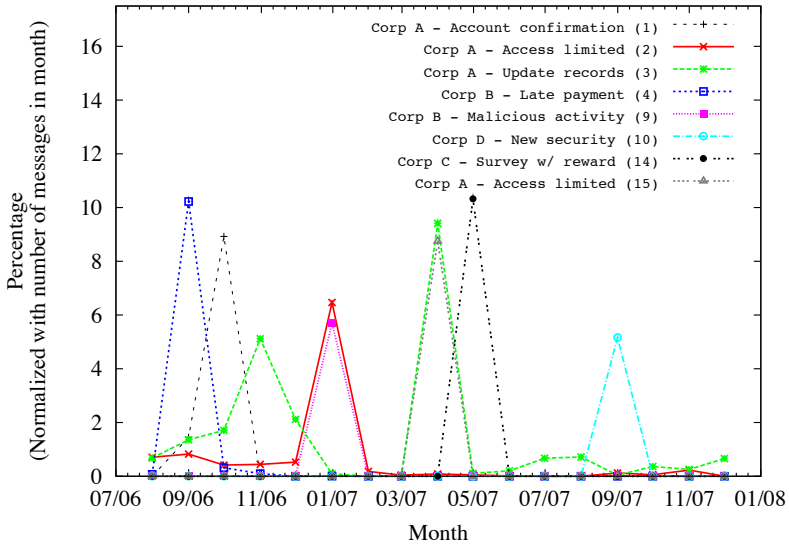


Figure 6: Clusters from Top 15 exhibiting "Flash attack" behavior

Table 3: Five largest clusters

Description	***Messages***
Corp A - Account confirmation	4091
Corp A - Access limited	3849
Corp A - Update records	3701
Corp B - Late payment	2948
Corp A - Verify account	2705

3.3.3 Discussion

We see from our results that phishing messages have a significant amount of duplication – averaging 64.1% and ranging from 36.7% to 81.4% on a month to month basis. We also found that for phishing messages categorized as flash attacks, there is a large amount of duplication in a short period of time. To filter these kinds of phishing messages a content-based collaborative filter would have to be responsive enough to detect these quickly. For phishing messages categorized as non-flash attacks, the amount of duplication is spread out over a much longer period, and a content-based collaborative filter would be very effective in filtering these kinds of messages.

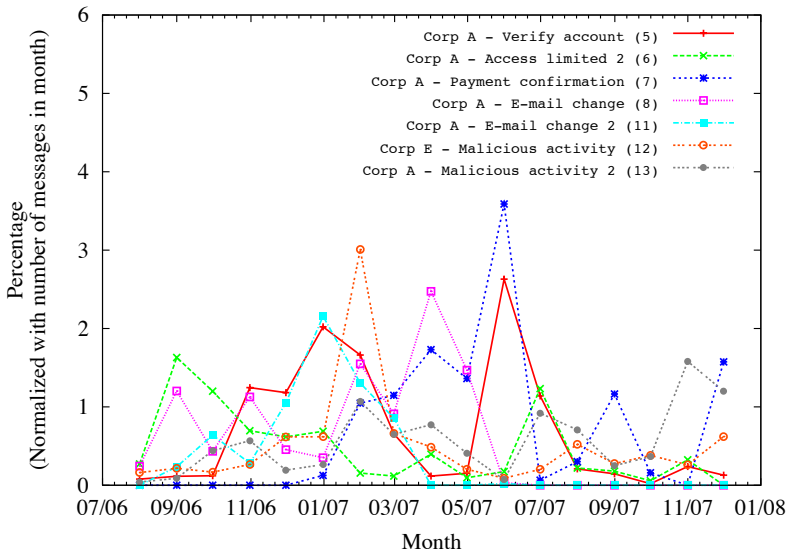


Figure 7: Phishing clusters from Top 15 exhibiting "Non-flash attack" behavior

3.4 Phishing Features Characterization

Having studied the content of phishing messages, we next answered similar questions about the construction techniques or characteristics of a message. Specifically we wanted to answer:

- Is a feature transitory (used only once or twice), or is it pervasive (used in many clusters)?
- If it was used in many clusters, what was its lifetime?

3.4.1 Features

A feature can be any construction technique or characteristic of a phishing message that can be identified by running a test over the raw message. We represent a feature with a mnemonic name using Java variable naming convention. An example of a feature is htmlMessage, which identifies a message containing HTML; Table 4 lists additional example features with brief descriptions. Most of the features we used are binary in nature, but we also had numeric and text features. Binary features are used to identify the result of a binary feature in the message. For example, the binary test scriptInHtml is a check of whether the HTML contains the "script" tag. Numeric features can be decimal values,

Table 4: List of selected features with brief descriptions

Feature Name	***Description***
forgedMuaOutlook	Forged mail pretending to be from MS Outlook.
hideWinStatus	Hide actual link by changing the displayed URL using "onmouseover".
htmlMessage	Message contains HTML.
htmlMimeNoHtmlTag	HTML message, but no HTML tag.
htmlTagBalanceBody	HTML message has unbalanced body tags.
mimeBoundDigits	Spam tool pattern (contains digits only) in MIME boundary.
mimeHtmlOnly	Message only has text/html MIME parts.
mimeHtmlOnlyMulti	Multipart message has text/html MIME parts only.
missingMimeOle	Message has X-MSMail-Priority, but no X-MimeOLE.
msgidFromMtaHeader	Message-ID from MTA header.
msgidSpamCaps	Spam tool Message-ID (CAPS).
mpartAltDiff	HTML and text parts are different.
msoeMidWrongCase	MS Outlook Express Message-ID wrong case.
normalHttpToIp	Dotted-decimal IP address in URL.
partCidStock	Spammy image attachment (by stock Content-ID).
scriptInHtml	HTML contains "script" tag.
phSubjAccountsPost	Subject contains particular words (Activate, Verify, Restore, Flagged, Limited, ...).
phRec	Message has standard phishing phrase "your account record".
urlHex	URL contains Hex characters.
weirdPort	Non-standard port number for HTTP.

or they can be counts. For example, the numeric feature htmlImageRatio has a decimal value that identifies the ratio of text to image area in a message, and the numeric feature numMimeParts has a count value, which identifies the number of MIME parts in a message. Text features are used in cases where we can run some post-processing and grouping over the feature. mimePartFileResult is an example of a text feature that stores the result of the linux command "file" on each MIME part of a message.

In order to have a comprehensive set of features, we defined 97 of our own features in addition to adopting over 600 features from SpamAssassin's non-network tests[3]. We do not describe all the features in detail, but instead describe the major feature groups and provide an example for each grouping (shown in Table 5).

[3]We used SpamAssassin 3.2.4, which was the latest version as of March 2008

Table 5: Features broadly split into four groups

Group Name	Description	Example
Header features	Features for tests that are run over the headers of the phishing message.	msgidSpamCaps is a binary feature that identifies a message where the Message-Id contains capital characters (indicative of a fake Message-Id).
Content features	Features for tests that are run over the content of the phishing message.	mimeHtmlOnly is a binary feature that identifies a message which has only HTML MIME parts.
URL features	Features for tests that are run over the URLs in HTML and text parts of a message.	numDomainDots is a numeric feature that is a count of the number of dots in the domain part of the URL.
Meta features	Features which represent a combination of features.	ppPhish is a binary feature that identifies a message as a PayPal phishing message, if it is from PayPal (fromPayPal) and contains an IP address in the URL (normalHttpToIp).

3.4.2 Characterizing features

To start, we identified all the features in each cluster. This was done by running all the tests over the messages belonging to the cluster. We then grouped together all the flash attack and non-flash attack clusters containing a particular feature and separated the features into two categories: transitory features and pervasive features.

For each feature we graphed the percentage of the number of messages within a cluster (on the y-axis) that satisfied a certain condition (i.e. positive test for a binary feature; a numeric feature above a certain threshold) against the period of data collection (on the x-axis). The graphs (Figure 8 to Figure 9(a)) in this chapter follow the same format and include only flash attack clusters. This is mainly due to the relatively small overlap between clusters, reducing clutter and making the graphs far more readable. Further, to reduce the noise in the graphs, we omitted a data point if it represented less than 1% of the number of messages in that month (typically toward the beginning or the end of the flash attack cluster's lifetime).

Transitory features are characterized as being used in only one or few clusters that occur within a short period of each other. Table 6 shows the transitory features in groups similar

Table 6: Transitory features

Group Name	**Features of Flash Attacks**	**Features of Non-Flash Attacks**	**Example**
Header features	phSubjAccountsPost msoeMidWrongCase mimeBoundDigits msgidSpamCaps missingMimeOle	phSubjAccountsPost msoeMidWrongCase mimeBoundDigits msgidSpamCaps missingMimeOle	Figure 8(a) shows phSubjAccountsPost is used in 96% of the "PayPal - Access limited (2)" cluster.
Content features	htmlMimeNoHtmlTag mimeHtmlOnlyMulti mpartAltDiff scriptInHtml htmlTagBalanceBody partCidStock	htmlMimeNoHtmlTag mimeHtmlOnlyMulti mpartAltDiff scriptInHtml htmlExtraClose htmlImageOnly32	Figure 8(b) shows htmlMimeNoHtmlTag used in 100% of the "eBay - Mailicious activity (9)" cluster.
URL features	normalHttpToIp hideWinStatus urlHex weirdPort hideWinStatus	normalHttpToIp hideWinStatus urlHex weirdPort	Figure 8(c) shows normalHttpToIp used in 100%, 98%, and 97% of the "PayPal - Account confirmation (1)", "PayPal - Access limited (2)", and "eBay - Late payment (4)" clusters respectively.
Meta features	None significant	None significant	Not applicable

to those used by the grouping of features earlier. Overall, 9 of the 12 clusters have transitory features, which shows that not only does the content of a phishing message change, but in most cases the features vary as well.

Pervasive features are characterized as being used in many clusters and have a relatively long lifetime. Table 7 shows the pervasive features in groups similar to those used by the grouping of features earlier. Overall, we see that each pervasive feature is used in a large number of clusters and in some cases in all the clusters.

3.4.3 Discussion

From our results we see that phishing messages have features which are both pervasive and transitory. We found most of the transitory features of phishing to be strong indicators of a phishing message (the feature is not used in legitimate messages) which might indicate the reason these features are transitory. Further we find that the pervasive features are mostly weak indicators of a phishing message (the feature appears in legitimate messages

Table 7: Pervasive features

Group Name	**Features**	**Example**
Header features	forgedMuaOutlook msgidFromMtaHeader	Figure 9(a) shows forgedMuaOutlook used in 4 flash attack clusters over a period of 9 months and all 7 non-flash attack clusters over a period of 17 months.
Content feature	htmlMessage mimeHtmlOnly	Figure 9(b) shows mimeHtmlOnly used in 6 flash attack clusters over a period of 9 months and all 7 non-flash attack clusters over a period of 17 months.
URL features	None significant	Not applicable.
Meta features	None significant	Not applicable.

as well). This suggests phishing message producers limit the utility of transitory features in time (by avoiding them in future generations of phishing) and limit the utility of pervasive features by choosing features that also appear in legitimate messages. The results suggest that classification techniques which use features will need to be constantly updated with newer features or find strong indicators of phishing that are less transitory.

3.5 Related Work

Previous studies on the evolution of phishing have been mainly focused on the meta-content of phishing messages. The Anti-Phishing Work Group (APWG) publishes monthly trend reports (e.g. [8]) that consider organizations targeted (e.g. PayPal, eBay), countries in which phishing websites are hosted and volume. Similarly, Marshal published two security threat reports during 2007 [9, 10] which looks at similar meta-content characteristics of phishing (organizations targeted, country of origin and volume) were discussed. Interestingly, they do identify a particular group as being responsible for over half of all phishing messages but do not go into details or analyze the techniques used in the group's phishing messages. RSA [11] also looked at trends in the meta-content of phishing messages, but also identified emerging techniques being used by phishers.

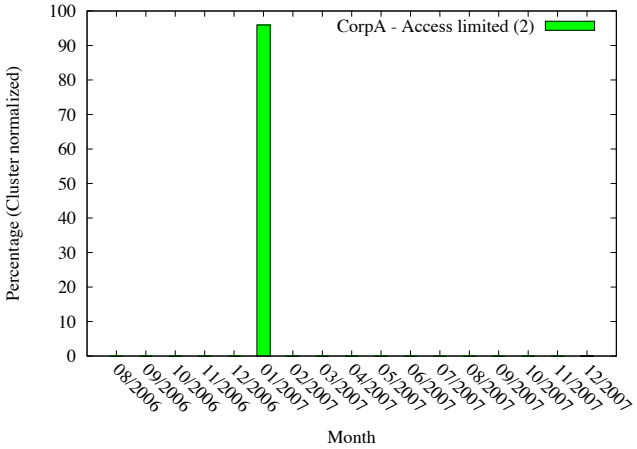
Beardsley [20] deconstructed and manually analyzed an attack from 2004. He observed some advanced elements of the phishing attack like cross-site scripting (XSS) and other elements which are "common to virtually all phishing scams".

Our study differs from the previous work on the evolution of phishing in several ways. First, we study unique clusters of messages and how they evolve, in terms of content and construction techniques used in creating the clusters. Second, our study analyzes 382,377 phishing messages over a 15 month period to produce clear and tangible evidence of evolution.

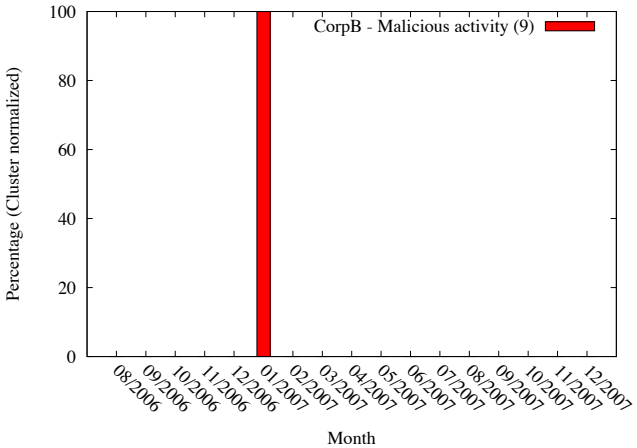
3.6 Conclusion

We analyze the evolution of phishing message content and phishing construction techniques in a large dataset of phishing messages (more than 380,000 messages over a 15 months period). From the content similarity point of view, the phishing messages are grouped into identifiable "attack campaigns" using a shingling algorithm as similarity test. These attack campaigns are then classified as either a flash attacks (a large volume of phishing messages sent in a short time) or non-flash attacks (similar messages spread over a longer period of time). Similarly, we group phishing construction techniques into: *transitory* features that are associated with a few clusters and are short lived; and *pervasive* features that are associated with a large number of clusters and have a long life span.

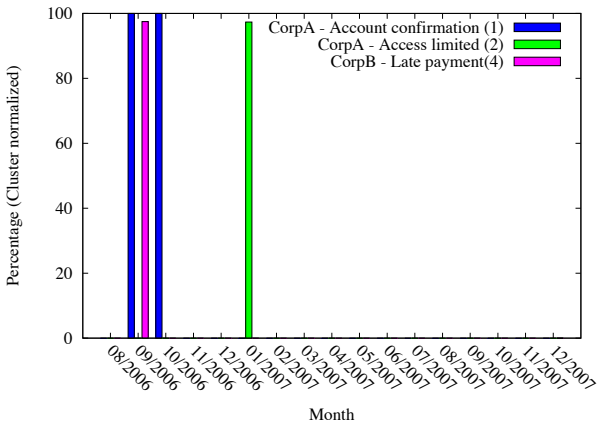
Our results provide a better understanding of phishing messages, both in terms of attack behavior and phishing identification features. However, our study also illustrates the need for further research on phishing. The large amount of duplication could be cause to look into signature-based content filters. Also, the strong indicators of phishing turned out to be transitory and therefore have limited utility for email filters. The pervasive features also have limited utility due the presence of these features in legitimate messages. These findings show the adaptive construction of phishing messages in each attack, carefully removing the transitory features that cause their identification by email filters. A serious challenge in phishing research is finding strong indicators of phishing that are less transitory.



(a) Transitory header features

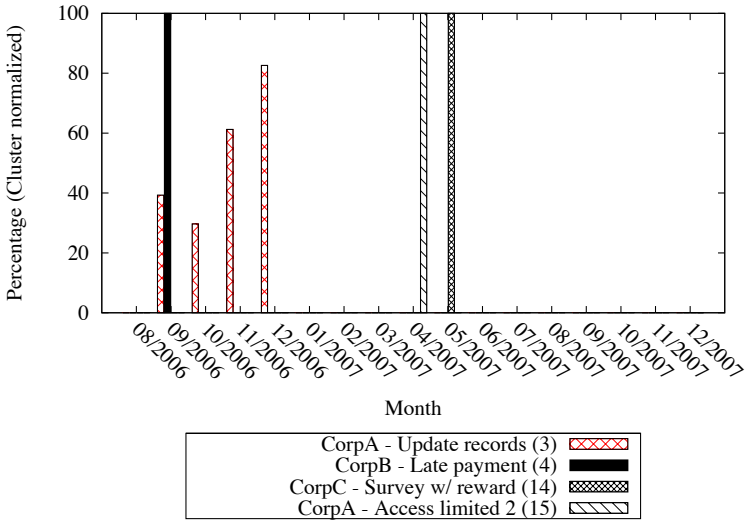


(b) Transitory content features

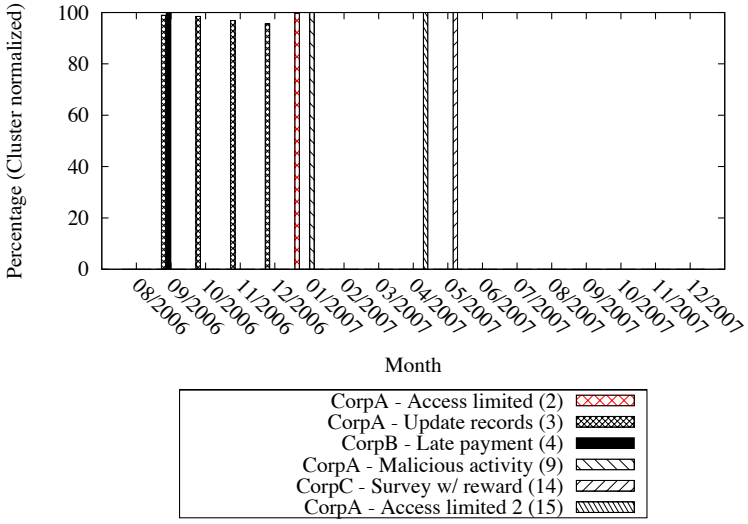


(c) Transitory URL features

Figure 8: Illustration of transitory features in different groups.



(a) Pervasive header feature



(b) Pervasive content feature

Figure 9: Illustration of pervasive features in different groups.

CHAPTER IV

DETECTION OF SOCIAL SPAM PROFILES USING STATIC CONTENT

Social networks have grown to become an important part of social interactions for personal and business reasons. Consequently, spammers have targeted social networks as media for propagating spam. Unlike email, interactions within current social networks such as MySpace and Facebook are restricted to members of the same social network. Consequently, spam must originate from a profile inside the social network. Although the maintenance of such rogue profiles (usually called "spam profiles") is a deterrent, it has not stopped the proliferation of spam in social networks [101].

Anecdotal evidence indicates that techniques used by social network operators to detect rogue profiles in practice include collaborative filtering and behavioral analysis. Dynamic methods such as collaborative filtering (where users vote on the nature of profiles) and behavioral analysis (where logs of interactions are used to detect spamming patterns) may be eventually able to detect rogue profiles, but they require a non-trivial amount of lag time to accumulate sufficient evidence. In this chapter, we study the analysis of static user profile content, which complements the dynamic methods. The main advantage of using machine learning analysis is that it can be applied as soon as the profiles are created, thus detecting suspect spam profiles before they have active propagated spam. In compensation, analysis based on machine learning depends on the quality of training data for an accurate prediction.

This chapter explores the limits and potential of machine learning analysis on static profile content at creation time (or within a small time period). To test the limitations on the accuracy of such analysis, we collected a large corpus of 1.9 million MySpace profiles, including approximately 1,500 confirmed spam profiles [101]. Our analysis considers two kinds of data in a social network profile:

- Categorical data - fields that can take only a limited number of values, for example: "Sex", "Age", and "Relationship Status".

- Free-form data - usually text information entered by users, for example: "About me" and "Interests".

This distinction allows appropriate machine learning algorithms to be applied to different sections of the data (e.g., decisions trees for categorical data and naïve Bayes for free-form data). To measure the discriminatory power of features and find the most important features in the identification of spam profiles, we apply the χ^2 test to measure the correlation (or lack of independence) between the features being studied and predicted class (whether a profile is rogue).

We also compare supervised machine learning techniques in their ability to detect spam profiles. On categorical data, the classifiers tested are: (1) AdaBoost algorithm (with a DecisionStump weak-classifier), (2) C4.5 decision tree, (3) Support Vector Machine, and (4) a neural network-based algorithm. In addition, we use naïve Bayes on the free-form data to classify rogue profiles.

Lastly, we perform a sensitivity analysis of the classifiers with respect to the evolution of spam profile content. Evolution of spam content is due to adversarial classification, where detection measures adopted by a social network can be countered with spammer's adaptive modification of the spam profiles to escape detection. Spam content evolution is a well known and documented phenomenon [46, 81]. We evaluate the robustness of classifiers by simulating adversarial action (e.g., removing the highest discriminative features) and re-evaluating the effectiveness of classifiers under the new assumptions.

Our results show that analysis of static profile content based on machine learning methods has the potential to improve significantly the detection of rogue profiles in social networks with non-trivial user profile content (e.g., MySpace and Facebook). This is particularly the case for recently created rogue profiles that have not engaged in spamming activity. Since this analysis complements dynamic methods, it has good potential for practical impact.

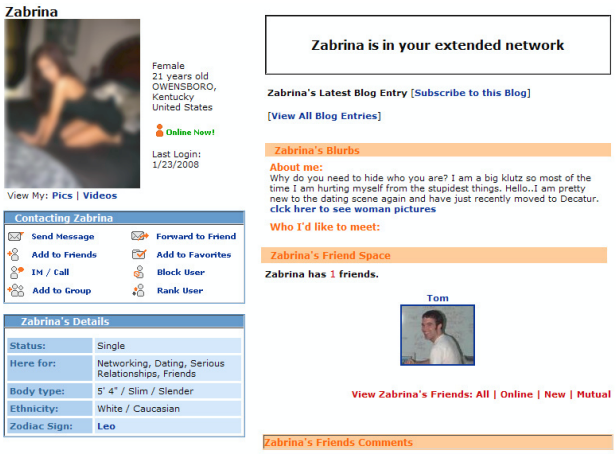


Figure 10: Example of a social spam profile

4.1 Social Spam Profiles

Social networks allow users to meet new friends, keep in touch with old friends, and participate in various social interactions. A user's presence in a social networking site is represented by a *social profile*, which allows him to maintain an identity within a social network and participate in it.

Spammers looking to propagate spam through a social network need to create a social profile. Once this *social spam profile* profile is created, spam can easily be sent out to other users using mediums offered within the community (e.g., friend requests, messaging, and commenting). Such spam has already been seen in the wild [101], and previous work [100] has focused on gathering and characterizing such profiles.

An example of a MySpace social spam profile is shown[1] in Figure 10. The profile contains a large amount of personal information, including various deceptive properties. As typical of spam profiles, this profile portrays an attractive, young, single woman with a provocative image to entice users to view them. Once the profiles have attracted visitors, they direct them to perform an action on an external website, usually by providing an alluring story in their "About me" section. For example, the profile in Figure 10 provides a link to an external website to "see woman pictures".

Early and accurate detection of such spam profiles is essential to reducing the amount

[1]Provocative images have been blurred to avoid offending readers.

of spam on social networks. We explore the benefits and limitations of profile classification using static user content entered during profile creation (or modification) to determine whether a profile is spammy or not. Ideally, this technique would be accurate enough to allow zero-minute determination about whether a profile is spammy or not and prevent a spammer from gaining an entry point into the social network. In practice, we submit that this technique would most likely have to be used to lower or raise another technique's decision boundary, allowing it to come to a quicker decision.

Current techniques of social spam profile detection [69, 108], rely heavily on detecting the spamminess of artifacts created by a user profile. For example, they analyze messages, comments, and friend requests. This approach must wait for the social spam profile to impact the social network with spam. Also, depending on the discriminatory power of features based on such artifacts, a large number of artifacts may be required before a definitive decision can be made.

Some social network sites use collaborative filtering or administrator-based reporting to manually identify social spam profiles. These techniques also suffer from a lag time between profile creation and identification, which can result in spam already having impacted users. Additionally, collaborative filters require that the profiles attract enough votes from other users to be marked as spam. Higher requirements in the number of votes will result in longer certainty in the evaluation of spam with a longer time required to reach the threshold, and vice versa. The benefit to using these dynamic techniques is usually a higher accuracy.

4.2 Experiment Setup

MySpace is one of the largest and most popular social networks, making it a prime target for social spam. It also features a large amount of personal user content per profile, and most of the information is publicly viewable by default.

4.2.1 Data Collection

With over 200 million profiles on MySpace, collection of all the profiles would be infeasible due to computational, storage, and network loads. We previously collected, in June to September 2006, a small subset of profiles from MySpace using two different sampling

strategies:

- Top 8 - Starting with a seed list of random profiles, the top 8 most popular friends were crawled, and subsequently their top 8 most popular friends were crawled in a breath first search (BFS) manner. This resulted in a collection of 891,167 connected profiles.

- Random - Profiles were crawled by generating random userid's and retrieving the profile represented by that user. This resulted in a collection of 960,505 profiles.

More details regarding these crawls, including an analysis of demographic information and language model characterization, can be found in our previous work [26].

Spam profiles from MySpace were previously collected by setting up honeypot accounts and collecting profiles that sent friend requests to these accounts. Fifty-one identical honeypot accounts portraying a fictitious, young, single male were spread over geographical locations in the United States resulting in 1,496 spam profiles collected from October 2007 to February 2008. The details of the honeypots, the exact methodology used and a study of demographics of the spam profiles are in our previous work [100].

4.2.2 Datasets

To use supervised learning, we need to provide spam and legitimate (non-spam) labels for a training set of profiles. We assign labels based on the method of collection with profiles collected via the honeypots being marked as spam and profiles collected via the top 8 or random sampling strategies as non-spam. As it was possible that during the top 8 or random sampling some spam profiles may have been inadvertently crawled, we ran various heuristics to detect spam profiles within these collections. For example, we used features from some of the previous work [69,108] in our detection of spam profiles, and we looked for keywords in free-form text fields with external spammy-looking links. Furthermore, during our classification experiments, we paid close attention to any misclassification and manually verified some of the labels.

We created two datasets for classification based on the above features and labels: the *top*

Table 8: Subset of fields parsed from a MySpace profile and a brief description of non-obvious fields.

Field	***Field Type***	***Description***
Age	Categorical	
Gender	Categorical	
Marital Status	Categorical	
Smoke	Categorical	Does smoke?
Drink	Categorical	Does drink?
Kid	Categorical	Want kids?
Zodiac	Categorical	Zodiac sign
Education	Categorical	Level of education
Orientation	Categorical	Sexual orientation
About Me	Free-form text	

8 dataset includes a random sample of 15,000 non-spam (legitimate/ham) profiles from the top 8 sampling and all 1,496 spam profiles from the honeypot profiles; the *random dataset* includes a random sample of 15,000 legitimate profiles from the random sampling and 1,496 spam profiles from the honeypot profiles. The rationale for using a subset of all the data available is that as legitimate profiles are so dominant, a majority classifier (i.e. a classifier which picks the dominant label for all profiles), would achieve over a 99.8% accuracy rate.

4.2.3 Feature extraction

Using a custom parser, we extract fields from the MySpace profiles. A subset of these fields can be found in Table 8 along with a brief description and whether we treat the field as a categorical feature or a free-form text feature. Most of the categorical data can, with minimal processing, be used as categorical features for classification. Using a bag-of-words approach, a free-form text field is broken into multiple features (with each word being a feature) after the application of stemming and the removal of stop-words. More details on the features used can be found in Section 4.3.

4.3 Feature Analysis

Before looking at classification, we take a closer look at the zero-minute features available for classification. These features based on static profile content are evaluated on their discriminative power and their robustness to adversarial classification.

Discriminative power of a feature can be seen as how strong of a signal it provides in

determining the resulting class. To compare the discriminative power of features, we use the χ^2 test, which measures the lack of independence between a feature f and the class c. The χ^2 test takes into account values from a two-way contingency table between f and c, which represents the number of times f and c co-occur, the number of times f occurs without c, and so-forth. Higher values of the χ^2 test indicate a stronger discriminative power.

Robustness to adversarial classification, to a certain extent, is a subjective measure that we manually assign. It is based upon how easy or difficult it would be for an adversary to change a feature in a spam profile in order to evade a classifier. To evade a classifier, the features need to blend in with the distribution of non-spam profile values without reducing the effectiveness of the spam profile. This can be seen as how easy it is for an adversary to "switch off" a signal.

The reason we use these two characteristics to distinguish features is that a highly discriminatory feature with low robustness would yield very good classification results, but it would not be very useful over a long period of time against an adversary. This is because a feature that is not robust could be easily removed by an adversary and would likely degrade the classification results due to its high discriminatory power. Ideal features would have both a high discriminative power and high robustness to adversarial classification.

4.3.1 Categorical features

Categorical features are obtained by extracting values from the respective categorical fields. Categorical fields with text values such as "Gender" and "Orientation" are converted to nominal values simply by assigning each distinct text value a nominal counterpart. Categorical fields with numeric values such as "Age" are left as numeric features. Two fields which are treated in a binary fashion are "IsPrivate" and "DefaultIM" because the only possible values for those fields are true or false. Although fields like "Smoke", "Drink", and "Kid" could also be treated in this manner, we choose to make a distinction between whether or not those fields are unanswered by assigning a value of "undef" if the field is not answered. If a numeric field is unanswered, it is assigned a value of "-1".

4.3.1.1 Discriminatory power

Figure 11(a) shows the results of the χ^2 test on the categorical features for the random and top 8 datasets. Each feature is represented by a cluster of two bars on the x-axis (one bar for each dataset), and the y-axis represents the χ^2 test score.

The features with the highest overall discriminatory power for both the random and top 8 datasets were "Kid", "Smoke", "Drink", and "DefaultIM". The "Kid", "Smoke", and "Drink" features are highly discriminative because over 95% of the spam profiles left this value blank (i.e., they have a value of "undef"), whereas only 15-25% of the non-spam profiles in the random dataset and 35-50% of the non-spam profiles in the top 8 dataset had these fields blank. The fewer non-spam profiles in the random dataset having this blank explains why the discriminatory power was higher as compared to the top 8 dataset. The "DefaultIM" feature had a very high discriminatory power in the top 8 dataset because 1% of the non-spam profiles left the field blank, whereas the spam profiles and the random dataset non-spam profiles had a mix of having a blank value or being filled out. It is surprising to see the "Age" feature as a good discriminator. However, we found that most non-spam profiles in the random dataset are under the age of 30 (with a peak at 17 years), and non-spam profiles in the top 8 dataset are under the age of 33 (with a peak at 18 years). The spam profiles, on the other hand, mimic profiles of young women with ages between 17 and 34 and a peak at 24 years (with over 85% between the age of 21 to 27).

Categorical features with the lowest overall discriminatory power for both the random and top 8 datasets were "Zodiac" and "IsPrivate". The reason "Zodiac" has a very low discriminatory power was that it seemed that the spam profiles had randomly chosen a zodiac sign with approximately 8% of profiles falling into each of the zodiac star signs. All the spam profiles had "IsPrivate" set as false, which coincided mostly with the non-spam profiles in the random and top 8 datasets.

4.3.1.2 Robustness of features

As the categorical features are simply values of fields, most of them can be easily changed. The robustness, in this case, mainly comes from if changing a value of a field, and thereby

value of a feature, would make the spam profile less effective (i.e., getting fewer users to click on a link). As mentioned in Section 4.1, most spam profiles portray young, single women and present other characteristics that make them more likely to be successful in spamming. Thus, the features "Age", "Gender", and "Status", would have a high robustness as they must take on certain values to be effective. Examples of low robustness features are "Kid", "Smoke", "Drink", and "DefaultIM".

4.3.2 Free-form text features

To convert the free-form text fields into features, we combine all the free-form text fields and use a bag-of-words approach where each word is treated as a binary feature (present or not). Before being treated as a feature, each word is stemmed and checked if it is a stop-word. Porter's stemming algorithm [80] first reduces the words to their root, e.g., "jumping", "jumper", and "jumped" are all reduced to their root "jump". Stop-words are words commonly used in a natural language and do not contain any significant meaning. We remove these words as they can confuse classifiers due to their high occurrence. Examples of stop words are "a", "the", and "it".

We also removed words which had less than 20 occurrences (a tunable threshold) in the entire dataset, leaving us with 2,289 words and 3,535 words in the random and top 8 datasets respectively.

Not all profiles contained free-form text features because they were blank or private profiles, and a few contained unparsable free-form fields. In the random dataset, approximately 9000 non-spam profiles (55%) did not contain any text in the free-form fields; similarly, in the top 8 dataset, approximately 4300 non-spam profiles (26%) did not contain any text in the free-form fields. There were 789 spam profiles which did not contain any text in the free-form fields.

4.3.2.1 Discriminatory power

Once again, we use the χ^2 test to find the most discriminative features separating the spam and non-spam classes. Figure 11(b) shows some of the words with the highest and lowest

discriminatory power using the χ^2 test. As expected, the words with the highest discriminative power are spammy words because these words likely occur in most spam profile free-form text but not in legitimate profile free-form text. A few of the non-spam words that do occur in the most discriminative top 20 words are words used in conjunction with spammy terms such as [check out my] "picture", "watch" [pictures/videos], and "heard" [of a drug].

Although the individual χ^2 test scores of the most powerful discriminative free-form text features is lower than the categorical, we have over 2,000 free-form text features present for each dataset. Some features might be co-dependent, but we assume independence as this a standard assumption when dealing with text classification.

4.3.2.2 Robustness of features

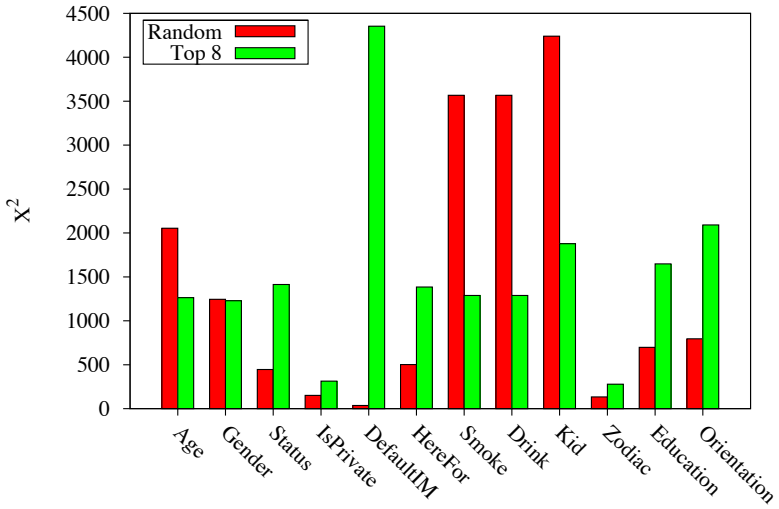
As previously mentioned, as most of the free-form text features with the highest discriminatory power are spammy words, an adversary could change the free-form text to not include spam words. This is not very practical as: a) removing spam tokens from the free-form text fields would make the spam profiles less effective, and b) to be most effective, the free-form text fields must be well written, which requires spammers to manually rewrite the text. Although an adversary might attempt to replace the words with the highest discriminatory power with a synonym or another form of camouflage (e.g. replacing "click" with "c1ick"), our detection techniques could similarly be tuned to detect synonyms and adjust for this.

To be conservative, we assign 30% of the most discriminatory words a low robustness, as even with the low practicality for the adversary, these words are most likely to be changed first. Words after this are assigned a medium to high robustness.

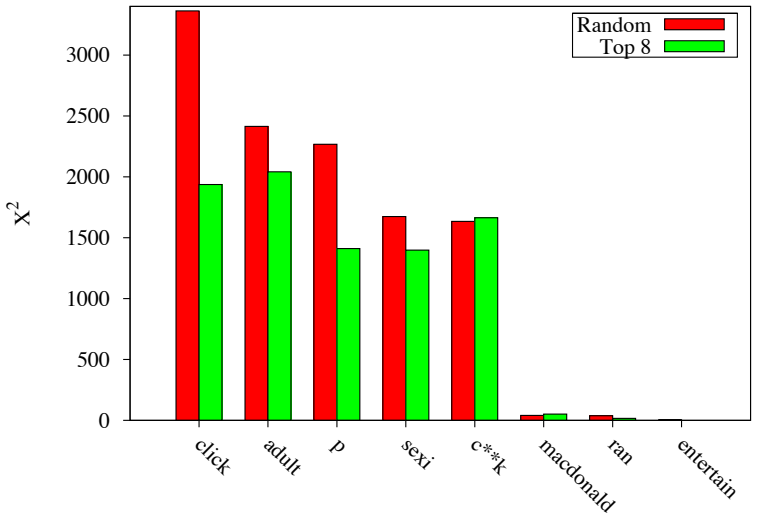
4.4 Analysis of Classification Results

We separately evaluate the effectiveness of using categorical and free-form text features, followed by an evaluation using a combined classifier. As a baseline, we compare our results to a basic majority classifier, which on both datasets results in a 90.9% accuracy with an AUC of 0.499. The majority classifier, which simply classifies all instances as the class with the most instances, classifies all profiles as non-spam, which results in 1496 (9.1%)

false-positives and 0 false-negatives.



(a) χ^2 test for categorical features



(b) χ^2 test for a sample of free-form text features

Figure 11: Discriminatory power of features measured by the χ^2 test for both categorical features and free-form text features.

Table 9: Results of classification based on the different feature sets.

Classifier	***Feature Set***	***AUC***		***False-Positives***		***False-Negatives***	
		Random	Top 8	Random	Top 8	Random	Top 8
Majority Classifier (Baseline)	N/A	0.499	0.499	1496	1496	0	0
AdaBoostM1 (w/DecisionStumps)	Categorical	0.991	0.982	98	1347	151	4
J48	Categorical	**0.994**	**0.994**	**24**	**23**	**55**	**85**
SMO (w/PolyKernel)	Categorical	0.986	0.984	34	34	66	136
Multilayer Perceptron (MLP)	Categorical	0.994	0.992	24	37	48	98
Naïve Bayes	Free-form text	0.718	0.886	849	866	635	71

Table 10: Results of classification under assumption of an adversary. Features with high discriminatory power and low robustness are removed.

Classifier	***Feature Set***	***AUC***		***False-Positives***		***False-Negatives***	
		Random	Top 8	Random	Top 8	Random	Top 8
Majority Classifier (Baseline)	N/A	0.499	0.499	1496	1496	0	0
AdaBoostM1 (w/DecisionStumps)	Categorical	0.916	0.980	1496	1496	0	0
J48	Categorical	**0.942**	**0.987**	**219**	**39**	**953**	**225**
SMO (w/PolyKernel)	Categorical	0.500	0.965	1496	76	0	285
Multilayer Perceptron (MLP)	Categorical	0.912	0.988	950	53	807	237
Naïve Bayes	Free-form text	0.718	0.886	848	863	612	67

Table 9 shows the result of classification of social spam profiles on the random and top 8 datasets. Results in bold highlight the best results for a particular classifier and measure (averaged over both datasets).

4.4.1 Categorical features

We first look at the results of the classifiers that use only the categorical set of features. Based on Table 9, we see that the J48 classifier performs the best, misclassifying only 79 and 108 profiles (accuracy of 99.6% and 99.4%) on the random and top 8 dataset respectively. This is followed closely by MLPs. SMO (with PolyKernel) and AdaBoostM1 (with DecisionStumps) perform well in terms of AUC but have a lot of FPs and FNs, comparatively.

J48 performs the best as there are some very discriminatory features, which allow it to quickly detect "obvious" spam or non-spam in the early levels of the tree, followed by refinement at the lower levels. As an example, the root of the J48 tree for the random dataset simply checks if "Smoke" is "undef" and if not, detects the instance as non-spam. This correctly classifies 78% (11,734 of 15,000) non-spam profiles with an error of less than 1%. For the same reason, we expected AdaBoostM1 (with DecisionStumps) to perform much better than it did. DecisionStumps are simply single branch trees, and they should have been able to classify the spam profiles even using only 10 iterations of boosting. In fact, for the default ROC threshold used by Weka, the AdaBoostM1 (with DecisionStumps) classifier performs only marginally better than the baseline majority classifier in terms of the number of misclassified spam profiles on the top 8 dataset.

Most classifiers also tended to perform better on the random dataset than on the top 8 dataset. This can be explained by the profiles in the top 8 dataset being more similar to the spam profiles (based on the overall lower χ^2 test score), causing more misclassifications.

On investigating the misclassifications, we found that most of the false-positives were due to a particular "strain" of spam profiles, which were more complete and contained legitimate looking categorical features. Additionally, unlike other spam profiles, this strain seemed to be selling male enhancement drugs, and the profiles were in a relationship. It is likely that the classifiers attributed the strain of spam profiles as noise and avoided

over-training.

Empty or partially filled out profiles resulted in a small number of false-negatives for all the classifiers, followed by additional false-negatives for some classifiers that mistook partially filled out profiles of young women as spam. Some of the empty profiles were the result of profiles "Undergoing construction" at the point of crawling.

4.4.2 Free-form text features

We now explore the classification of profiles based on the free-form text-based sections of their profiles. From Table 9, we see that a naïve Bayes classifier using only free-form text features does poorly in comparison to the classifiers using only categorical features.

The large number of false-positives were due to 789 of 1496 spam profiles (53%) containing blank free-form text sections (or the free-form text sections not being parsable), which resulted in the majority non-spam label applied to such profiles. From a sample of profiles, we manually investigated and found the false-negatives were mainly due to non-spam profiles using spammy tokens in their about me section. For example, we looked at the most discriminatory spam features in a subset of non-spam profiles used in the random dataset; we found 47, 36, 457, and 90 occurrences of the spam tokens "click", "adult", "sexi", and "c**k", respectively.

4.4.3 Categorical and Free-form text features

A natural question which arises is whether the independent predictions of the classifiers on the categorical and free-form text features can be combined to improve our classification results. The aim here would be to use classifiers which are best at classifying their respective feature set (and allow for co-dependence in the case of free-form text features).

To do this we experimentally examine the results of applying the 'AND' and 'OR' operators to the predictions of the classification based on the default ROC threshold. When classifying spam profiles, in the case of the 'AND' operator, only if both categorical and free-form text classifiers predicted the profile to be "spam" will the profile to be marked as spam. In the case of the 'OR' operator, either categorical or free-form text feature classifier predicting the profile as "spam" will result in the profile being marked as such. To reduce

the number of false-negatives, we only considered the outcome of the free-form text classifier if the text feature set was not empty.

Intuitively, the 'AND' operator should reduce the number of false-negatives as both classifiers will have to predict a "spam" classification before one is assigned. The 'OR' operator on the other hand should reduce the number of false-positives as either classifier predicting "spam" will cause "spam" to be assigned. On the flip-side, the 'AND' operator will increase the number of false-positives due to more "spam" being classified as "non-spam" and similarly the 'OR' operator will increase the number of false-negatives.

We do not show results here because, as intuitively explained, applying the above operators to predictions simply reduced either false-negatives or false-positives but not both simultaneously. We hypothesize that building a new classifier over the combined categorical and free-form text features would alleviate this problem.

4.5 Analysis of Adversarial Classification Results

Previous evolutionary studies [46, 81] have shown that once techniques are developed to counter particular types of spam, spammers evolve and deploy new types of spam to bypass those techniques. Assuming the presence of an adversary with an ability to probe our classifier for most discriminatory features, we evaluate the effectiveness of our classifiers with the low robustness and high discriminatory power features removed—as adversaries are likely to remove most spammy easy-to-change (low-impact on spam) features first. To emulate this, we use the χ^2 test score of the features over both datasets and remove the highest discriminatory features with low robustness.

As a baseline, we again use a basic majority classifier, which on both datasets results in a 90.9% accuracy with an AUC of 0.499. The majority classifier classifies all profiles as non-spam, which results in 1496 (9.1%) false-positives and 0 false-negatives—the results remain the same as we do not change the number of profiles, only reduce the set of features.

Table 10 shows the result of classification of social spam profiles, assuming an adversary, on the random and top 8 datasets. Results in bold highlight the best results for a particular classifier and measure (averaged over datasets).

4.5.1 Categorical features

For the categorical set of features, the four features with the strongest discriminatory power and low robustness are "DefaultIM", "Smoke", "Drink", and "Kid". Based on Table 10, we see that the J48 classifier performs the best, misclassifying 1172 and 264 profiles (accuracy of 92.9% and 98.4%) on the random and top 8 dataset respectively. AdaBoostM1 (with DecisionStumps) and MLPs, also do well based on the resulting AUC, but AdaBoostM1 (with DecisionStumps) has a significantly higher number of misclassifications. SMO (with PolyKernel) performs the poorest, with the classification on the random dataset being similar to that of the baseline majority classifier.

J48 performs the best, for reasons similar to it doing the best at regular classification as well—it is able to use the highest remaining discriminatory features to detect "obvious" spam or non-spam in the early levels of the tree. As the most discriminatory features have been removed, the J48 decision tree resulting from this classification is more complex with the number of leaves in the tree growing by about 45%. Even with the best classification results, the J48 classifier misclassifies over 1400 profiles (4.3%) on both advesarial datasets, as compared to a misclassification of less than 200 profiles (0.6%) on both non-advesarial datasets. Although the lack of expressiveness of the PolyKernel is the likely reason for SMO with PolyKernel classifier performing badly on the random dataset, we are investigating this fruther.

On investigating the misclassifications we found that in addition to the false positives incurred by the non-adversarial classifier, profiles of women around the age of 30 started being marked as legitimate. Additional false negatives incurred by the adversarial classifier were due to partially complete profiles of women which had filled in certain categorical fields which previously would have identified them as non-spam, but which are no-longer considered.

4.5.2 Free-form text features

To pick which free-form text features to remove we averaged the discriminatory power of the free-form text features between datasets and chose features with the strongest discriminatory power and lowest robustness. We set a limit of disregarding 900 features per dataset, which left us with 1,389 features and 2,635 features in the random and top 8 datasets respectively. The results of running the classifiers over the new set of features, gives us the results shown in Table 10.

The naïve Bayes classifier performs similarly over the full feature set. This indicates that although the features with the strongest discriminatory power were removed, there is a large enough set of weaker features available for classification.

Once again, a large number of the false positives were due to spam profiles not containing a free-form text section, and the false negatives due to spammy tokens being used in legitimate profiles.

4.6 Related Work

Heymann et al. [44] provide an overview of social spam fighting mechanisms, categorizing them broadly into: Detection-, prevention-, and demotion-based strategies. They group classification approaches as detection-based strategies, likely due to the fact that previous work [69, 108] mostly makes use of features which require spammy behavior to be present, before classification is possible. We focus on a prevention-based strategy using machine learning, to identify social spam profiles before they can be used to propagate spam.

A lot of work has been done in the area of demotion [or promotion] strategies, namely trust. TrustRank [42] and SocialTrust [25] look at graph-based techniques to ranking nodes within a network. SocialTrust looks explicitly at trust in a social network, whereas TrustRank approaches the problem from a web spam perspective, both of which are modifications of the PageRank algorithm [79]. A specialized look into the trust of CouchSurfing.com and Del.icio.us is done by Lauterbach et al. [63] and Noll et al. [78], which use additional attributes between friends provided by the social network to facilitate trust calculations.

4.7 Conclusion

We explore the limits of classifying social spam profiles on MySpace. Specifically, we focus on zero-minute fields or static fields present when a profile is created to determine whether such a technique would be feasible in preventing spammers from gaining a foothold inside a social network to send spam. We start with an analysis of features that includes a statistical analysis to determine the discriminatory power of a feature as well as the robustness of features to adversarial attack. We then measure the effectiveness of classifiers that are built using these features.

Our classification results show that a standard C4.5 decision tree performs the best with an AUC of 0.994 and an average of approximately 24 false-positives and 70 false-negatives. The number of false-positives is acceptable as most of the non-spam profiles misclassified were partially filled out profiles. A subset of the false-positives are a new "strain" of spam profiles that expose one of the weaknesses of our classification technique—namely, a weakness in detecting new types of spam profiles without sufficient training examples. To further explore the potential of new types of spam profiles evading our classifiers, we use our earlier statistical analysis to disregard features that are most likely to be camouflaged by an adversary and analyze the effects this has on the classifier's performance.

We believe that the classification results are positive enough to justify building a system in which a classifier can automatically detect most spam profiles with a high confidence, and mark others as gray profiles. Collaborative filtering (with a lower threshold) or administrator reporting can be used to confirm the correct class label for these gray profiles, which in turn could be used in a feedback loop to improve future classification results.

CHAPTER V

DETECTION OF TREND-STUFFING ON TWITTER

Twitter is growing in popularity as a micro-blogging service for users to share a short message (called a *tweet* with a maximum length of 140 characters) with friends and others. It grew to over 7 million unique visitors [72] and 50 million tweets per day [102] in 2009. Current estimates put Twitter at 20 million unique visitors monthly and over 1.2 billion tweets a month [5]. Technically, Twitter is a multicast service, where *followers* subscribe to postings from users (and topics [91]) of interest. Twitter's homepage contains listings of *trending* topics under the headings of: currently popular, popular this hour, and popular today, as well as an abbreviated list alongside most user pages. Typically, trending topics are associated with news related to current events or upcoming events.

Trending topics help users receive the tweets they are interested in, and even allow visitors to catch up with the latest topics. However, such keywords can also be misused by spammers and other malicious parties (called *miscreants* for the lack of a better word) to promote their own interests, in a practice we call *trend stuffing*. Typical scenarios of misuse are tweets that include an unrelated, but popular trending topic. Such association can be achieved easily by adding the hashtag—a word or tag prefixed with a '#' character to identify topics—for a currently popular topic, to the spamming tweet. Unsuspecting followers of the trending topic will receive the tweet and may then click on the spammers websites, increasing their traffic and value. The practice of trend stuffing is against the rules outlined in the "Spam and Abuse" section of Twitter's fair use policy [4] and tools have been introduced to combat such tweets [3,31], but they have not been able to eliminate this practice yet. One plausible explanation for this persistence is that trend-stuffing breaks the rules and probably results in the suspension or deletion of the offending account, but new accounts can be easily created.

Automated identification of tweets that contain trend-stuffing is a significant challenge

due to the small amount of information contained within their 140-character limit. The first contribution of this chapter is an approach to automatically identify trend-stuffing in tweets. We do this in three steps: First, we build a model of tweet content associated to a trending topic. The goal of this model is to distinguish trend-stuffing tweets from legitimate tweets within the limited content information of a tweet. Second, when a tweet contains a URL, we use the URL to build a meta-model of its web page content. This meta-model is effective since most spam tweets aim to promote a website. Third, we combine the tweet content model with the web page content meta-model to increase its discriminating power.

The second contribution of the chapter is a quantitative evaluation of the performance of each model and their combination using a dataset of over 1.3 million tweets collected between November 2009 and February 2010. This evaluation data set includes the collection of webpages pointed to by the tweets, consisting of 1.3 million webpages following any redirect chains that may have been present. In addition, we also verified the longevity of tweet accounts that sent those tweets. If an account has been suspended, we assume that it has been classified as spammer or miscreant by Twitter. This information is used to sharpen our evaluation, for example, to verify if tweets classified as "not belonging to a trend" are actually trend-stuffed tweets. When using individual single-trend classifiers, the C4.5 decision tree classifier achieves the highest average F1-measure of 0.79 on the tweet text and 0.9 on the associated webpage content. Combining the classification predictions from the tweet text and the associated webpage content, we find that the C4.5 decision trees perform the best when combined with the 'OR' operator with a 0.9 combined classification. Combining the classifiers using simple 'OR' and 'AND' operators, allows us to take advantage of the performance of the respective classifiers, while "short-circuiting" the combined classification with a positive or negative prediction respectively.

5.1 Overview of Tweets and Trending Topics

Tweets (or statuses) are short 140 character messages that are posted by users in response to the question "What are you doing?". A user's tweets are delivered to all the friends following the user, and the tweets are also available on the user's account. As the evolution

of Twitter into an avenue for content-creation has steadily progressed, Twitter has changed the question posed to users (when posting a message) to "What's happening?" [91].

Trending topics on Twitter are popular topics that are being tweeted about the most frequently at a particular point in time. As the question being posed to users is now "What's happening?" popular topics are usually associated with current events. In fact, some trending topics have played a significant role in providing news for breaking stories and allowing users to provide opinions on current events (e.g., Haiti disaster relief). Twitter has also used hashtags—a word or tag prefixed with a '#' character—to identify such topics, although recently the requirement for a '#' before a tag has been dropped. We henceforth sometimes use *trend* as a short-form for a trending topic.

5.1.1 Trend Stuffing

Currently, trending topics are listed on the Twitter homepage as well alongside most other pages on Twitter. Due to their high visibility, trending topics attract tweets that may not be directly related to the topic—a practice we call **trend stuffing**. These deceptive tweets may arise from spammers, marketers, or users who want to promote a particular message (e.g., "Happy Birthday Tim #worldcup"). Twitter's Rules [4] forbid this behavior, stating permanent suspension will result from "post[ing] multiple unrelated updates to a trending or popular topic".

Twitter has taken steps to reduce the amount of noise in trending topics [3,31], and even though such measures have had an effect on the amount of obvious spam in the trending topics, it has not eliminated noise or spam completely. The exact details behind these many of these measures have not been released, but when details have been released, spammers have found a way around them. For example, to avoid the detection mechanisms that take into account the history of a user's tweets, spammers delete historic tweets (having only a few "non-deleted" tweets at a particular time). We further discuss some details of known measures used by Twitter in Section 5.5.

5.2 Tweet Classification

Our approach to trend stuffing can be split into three main steps. First, we study text-classification based only on the 140 characters (or less) of the tweet. This would indicate how well the text of the tweets themselves could be used to build a classification model for a trend. In addition as we will discuss in Section 5.3, using over 12,000 features for the classification model is not feasible and thus we needed to take an intermediary step to reduce the number of features considered for each trend.

Second, we look at text-classification based on webpages associated with a tweet. The intuition is that the content of the webpages associated with a tweet can also be used to determine if a tweet belongs to a trend or not. Further, this would likely be effective as since the intent of spammers is to promote a website, most spam tweets would contain a URL. Once again, the number of website content features is too large (over 500,000) to be feasible for classification, and thus we take an intermediary step of feature selection, discussed in Section 5.3, to reduce the number of features considered for each trend.

The third step involves combining the text-classification of both the tweet text and on the associated webpage content. In this case we compare the performance of using the 'AND' and 'OR' operator in combining the prediction from the tweet text classifier and associated webpage content classifier, in determining whether the tweet (and associated webpage) belong to the trend.

Before discussing the statistical methods used in our study, we describe the dataset used for our experiments.

5.2.1 Dataset collection

We gathered over 1.3 million tweets related to 2,067 trends over a 3 month period between November 22, 2009 and February 21, 2010. The top 10 most popular trending topics were queried every hour using the Twitter API [2], followed by subsequently fetching tweets associated with the top 10 trends every minute.

We also crawl all webpages linked by a tweet as tweets may contain little or no-text with only a link promoting a site. In fact, since we are particularly interested in finding

spammers or users who try to promote sites unrelated to a particular trend, we focus on tweets that contain links. We follow the link and its corresponding redirection chain to fetch the final webpage belonging to the link. This collection process generated over 1.3 million webpages, not including intermediate webpages responsible for redirection. Further details on how we handle redirect chains as well as JavaScript redirection can be found in our previous work [98].

The average size of a tweet in our dataset is 105 characters, with a standard deviation of 9 characters. The average size of a webpage linked to by a tweet is 55kB, with a standard deviation of 41kB. Although there are a number of other characteristics to explore, numerous papers have already focused on characterizing Twitter [45, 53, 59, 61]. In this chapter, we focus on the features used in classification and the actual classification itself.

We perform text-classification on both the tweet text and the content of the final webpage pointed to by the tweet. In both cases, we tokenize the text and use each word as a feature after performing Porter's stemming [80], removing stop-words, and removing tokens with less than 50 occurrences throughout the dataset. The tweet text also undergoes additional pre-processing to remove any text that is directly associated with the trending topic (e.g., trend hashtag or words found to be part of the topic). Finally, we count the number of occurrences of the token and use this count as a numeric feature. After performing these steps, the tweet text generated over 12,000 features, and the webpage content accounted for over 520,000 features.

5.2.1.1 Statistical Classification Setup

We use supervised learning because our initial experiments with an unsupervised learning algorithm (clustering using k-means and EM) generated unsatisfactory results. Specifically, we found the large number of features, and sometimes irrelevant features, caused the clustering algorithm to perform poorly. On the other hand, the supervised learning algorithms were able to quickly and accurately identify tweets not belonging to a given trend, even in the presence of noisy labels.

We build a classifier model for each of the 670 trends, with over 200 tweets, using one-to-many classification with all the tweets belonging to the trend in one class and a stratified sampling of tweets in the other trends (with at minimum one tweet per trend). As an example, if a particular trend has 500 tweets, we pick 669 tweets from the other trends resulting in classification for 1169 tweets.

We do not build one single multi-class classifier for multiple reasons: 1) retraining would be required for new classes (trends) and over time as topic drift occurs; 2) smaller classifiers would be quicker and more robust; 3) more aggressive feature selection can be performed to reduce the feature set to only those terms relevant to the particular trend.

As we are using supervised learning, we need to provide labels for training the classifier (or building the model). We use a label associated with each unique trend text (e.g., "welcome 2010" or "tiger woods") as assigned by Twitter. When assigning labels, we do not remove tweets from suspended accounts from the dataset because in practice, these algorithms will be trained on somewhat noisy data as tweets are only identified as "bad" after a certain time-lag. The classifiers will need to rely on the confidence bounds as well as cross-validation to only learn features of tweets related to the trend and mis-classify the noise. Although this decision might result in trends containing a large number of noisy labels such that the noise is included in the model of tweets belonging to the trend, this methodology represents a realistic scenario that we wish to analyze.

We use F1-measure to evaluate the performance of our classifiers. In our one-to-many experimental setup, positive tweets are tweets that belong to a given trend, and negative tweets are tweets that belong to other trends for the particular experiment. In our evaluation, we take into account the number of suspended tweets classified as positive which were labeled as positive but which should not have been—as mentioned earlier, suspended tweets are likely to not belong to any trend. Namely, the F1-measure is adjusted by taking into account the number of suspended tweets when calculating the precision and recall.

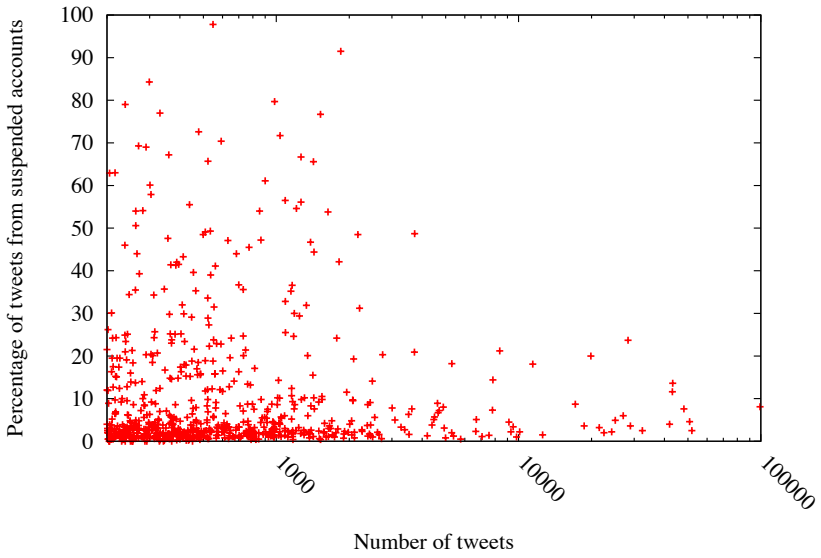


Figure 12: Percentage of suspended tweets vs number of tweets in a trend

5.2.1.2 Dataset Label Refinement

To identify tweets associated with trend stuffing, we re-queried the Twitter accounts of users who had tweets in our dataset to check if their Twitter account had been suspended. We found over 26,000 Twitter accounts associated with tweets had been suspended (or approximately 3.5% of all accounts collected), which were responsible for over 130,000 tweets (or approximately 10% of all tweets collected). We do submit that a very small percentage of these accounts may have been suspended for purposes other than spam, and in-addition, there are Twitter accounts which have been used for spam which have not yet been suspended by Twitter. Figure 12 shows the number of tweets in a trend on the x-axis, as well as the percentage of tweets in a trend which come from suspended accounts on the y-axis. Each point represents a unique trend.

5.3 Reduction of Feature Sets

Classification with over 12,000 features for the tweet text and over 520,000 features for webpage content is not practical. Most classifiers will perform poorly either due to the curse of dimensionality [21] or due to the computational burden of trying to analyze patterns associated with all the features (e.g., in Neural Networks learning weights for thousands of input nodes).

We chose Information Gain to reduce the number of features to a tractable size. Information Gain, from an information theory perspective, is a measure of the amount of expected reduction in entropy if one were to use the feature to partition tweets belonging to a trend, from tweets not belonging in a trend. After calculating the amount of Information Gain for each feature, we pick the strongest features.

Previous work on a study of feature selection in text categorization [104] found that Information Gain was effective in drastically reducing (by up to 98%) the number of features used for classification while not losing categorization accuracy. As we plan to do the feature selection for each trend when performing one-to-many classification, we believe an even more drastic reduction can be preformed.

We decided to evaluate our classification with a more tractable set of 100 and 1000 features for the tweet text—0.8% and 8% of the total number of tweet text features, respectively—and with a set of 100, 1000, and 5000 features for the webpage content—0.02%, 0.2% and 1% of the total number of webpage content features, respectively. Table 11 shows the average lowest Information Gain and the average total Information Gain. The average lowest Information Gain is the average Information Gain of the weakest feature in the feature set—it indicates that any features not added into the set will have an Information Gain value lower than it. The average total Information Gain can intuitively be thought of as the average total discriminative power of all the features used to distinguish the trend from others.

For the tweet text from the Table 11(a), we see that after 100 features, the average Information Gain of the weakest feature drops to 0.004, which is indicative of most of the discriminative power being captured in the first 100 features. This is further shown in Table 11(b), where the average total Information Gain increases by only 0.4 when considering an additional 900 tweet text features. In fact, we find that on average, after 317 features the Information Gain drops to zero. For evaluation purposes, we still consider tweet text classification both using 100 and 1000 features.

For the webpage content from Table 11(a), the average Information Gain of the weakest

Table 11: Comparison of the number of features considered, when using Information Gain to reduce the feature set size.

(a) Average lowest Information Gain

Feature category	***Number of features***		
	100	*1000*	*5000*
Tweet Text	0.004	≈ 0	-
Webpage Content	0.096	0.016	0.002

(b) Average total feature set's Information Gain

Feature category	***Number of features***		
	100	*1000*	*5000*
Tweet Text	2.1	2.5	-
Webpage Content	14.6	46.9	70.9

feature when considering 100, 1000, and 5000 features, is 0.096 to 0.016 to 0.002, respectively. Considering the average total Information Gain shown in Table 11(b), we see a 300% increase from 100 to 1000 features, and a further 50% increase from 1000 to 5000 features. The large increase in entropy when considering webpage content features, especially as large as when going from 100 to 1000 features, makes it likely that using 100 features for classification for webpage content may not be enough. We will evaluate webpage content classification using 100, 1000, and 5000 features.

5.4 Experimental Classification Results

We separately evaluate the effectiveness of classification on the tweet text and the webpage content, followed by an evaluation combining the predictions of both classifiers. As previously mentioned, we use tweets from suspended accounts to validate the false-negatives. Validating the true-positives requires a larger manual effort and we explore this for a subset of tweets in Section 5.4.4.

As a baseline for our classification results we use the majority classifier which classifies all profiles into the larger class. The class representing a sample of all tweets except tweets naïvely classified into the trend, has the same size as the opposing class, so both classes have an equal size. The majority classifier would have a 50% accuracy with a F1-measure of 0.66 (assuming the positive class is assigned the majority class) or 0 (assuming all tweets are majority classified into the negative class)—for our purposes we use a baseline F1-measure

of 0.66.

5.4.1 Classification using Tweet Text

We first look at the results of the classifiers that use only features based on tokens in the tweet text. The results of classification using these features is shown in Figure 13. Each set of bars represents a different classifier, and each bar within the set represents the average performance in terms of F1-measure on a different number of features. We see that naïve Bayes and C4.5 decision trees perform well, with C4.5 decision trees performing the best by a small margin. Decision Stumps on the other hand perform on quite poorly, worse than the baseline majority classifier.

C4.5 decision trees and naïve Bayes have the highest F1-measure as they have a lower number of false-negatives, or tweets classified as not being in the trend. Decision Stumps are only able to use a single feature to decide if a tweet is in the trend or not and thus in the absence of a "hashtag" is not able to perform very well. The poor performance of Decision Stumps was due to the low expressive power of a single Decision Stump and lack of a single feature that could, without noise, capture the essence of a trend. Compared to a sample of C4.5 decision trees, we found the root node to be the same, but the C4.5 decision tree had a number of branches under the root to filter out false-positives and false-negatives.

Figure 14 shows the F1-measure results of naïve Bayes classification, using 100 features. The number of tweets belonging to the trend being classified (the positive class) is shown on the x-axis with the F1-measure on the y-axis. We see that as the number of tweets in the positive class increases, the amount of variation in the F1-measure reduces significantly with the scores tending to be higher. The reason for this is, especially for smaller trends, some of the positive classes may contain a large percentage of noise (suspended tweets). If this is the case, the classifier learns the noise as part of the trend. In calculating the F1-measure, as we consider suspended tweets not part of the trend, this can reduce the resulting F1-measure. As the number of tweets in the positive class increases, the likelihood of noisy tweets being classified as part of the trend decreases, and thus results in fewer false-positives (i.e. tweets which were suspended and thus not part of the trend, being classified

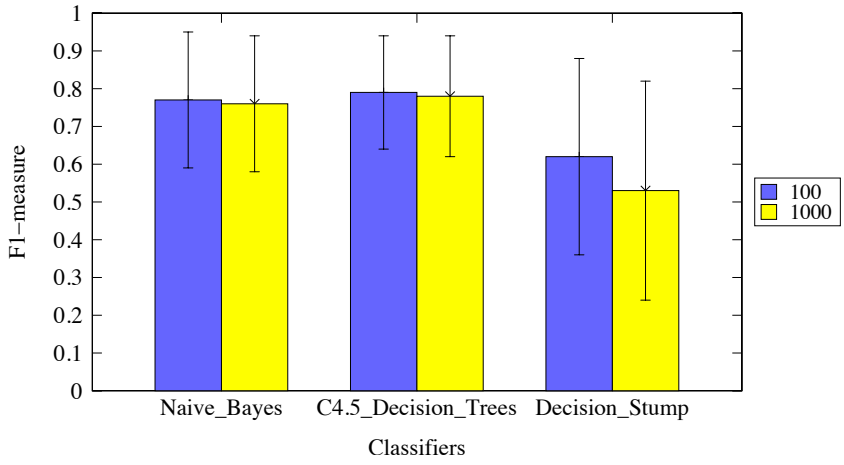


Figure 13: Results of classification on tweet text using different number of features. Error-bars represent one standard-deviation of the F1-measure.

as positive). The reason for the small difference between the classification using 100 features and 1000 features of tweet text, is likely due to the Information Gain per feature being very low after selecting the strongest 100 features (in some cases even zero).

One of the other important factors in performance of a classifier is the amount of time it takes to build the model and time to classify a tweet. Table 12 shows the result of the average model build time (training time) and average classification time (test time) for a single tweet for both 100 and 1000 features, using different classifiers. We see that Decision Stumps take the least time for both building the model and to classify a tweet, but this is due to the very simplistic single branch model. Naïve Bayes is faster at building a model than C4.5 Decision Trees as in naïve Bayes, each feature is treated independently and thus can be looked at once, whereas in when building a C4.5 decision tree, at every node, an attribute is picked based on the highest Information Gain of available features, followed by a recursion on the smaller sublists. C4.5 decisions trees are much faster than naïve Bayes on classifying a tweet once the model is built as a relatively small decision tree needs to be traversed to determine a label for the tweet, rather than naïve Bayes which needs to calculate the combined sum of probabilities for all the features. Naïve Bayes requiring to find the combined sum of probabilities, is the reason there is an over 400% increase in time taken when classifying a tweet with 100 features versus 1000 features.

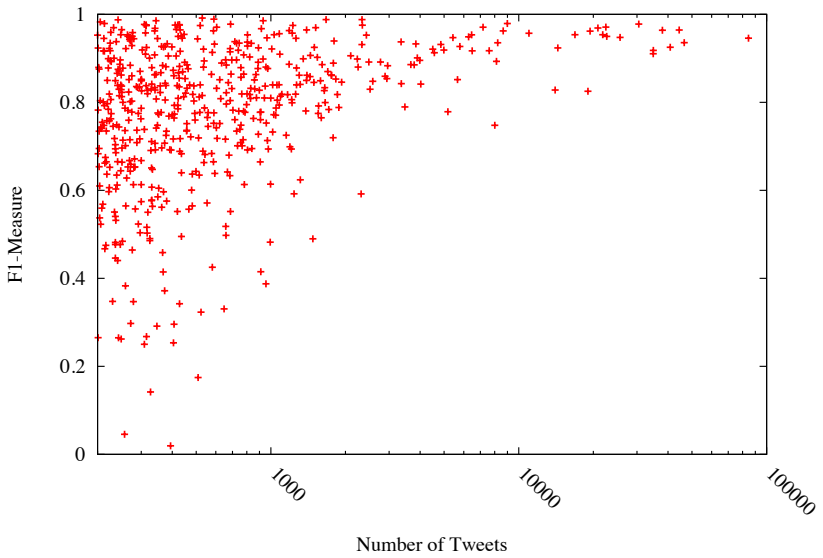


Figure 14: F-measure of naïve Bayes classifier using 100 tweet text features.

Table 12: Average time in ms, to train and test the classifiers over the tweet text.

Tweet text	***Number of features***	
Training Time (ms)	*100*	*1000*
Naïve Bayes	0.21	0.79
C4.5 Decision Tree	0.77	7.80
DecisionStump	0.22	0.86
Test Time (ms)	*100*	*1000*
Naïve Bayes	0.26	1.12
C4.5 Decision Tree	0.08	0.11
DecisionStump	0.07	0.11

5.4.2 Classification using Webpage Content

We now explore the classification of tweets based on the text-features from the webpage content associated with the tweets. Webpage content associated to tweets should also be related to the trend or be separable in such a way. The results of classification using these features is shown in Figure 15. Similar to Figure 13, each set of bars represents a different classifier, and each bar within a set represents the average performance in terms of F1-measure using a different number of features. The C4.5 decision trees perform the best, followed closely by Decision Stumps. Naïve Bayes performs the worst, but better than a baseline classifier.

C4.5 decision trees perform the best as they have the least false-negatives, followed by

the Decision Stump classifier. The tree-based classifiers (C4.5 decision trees and Decision Stump) perform well in this case as there are a few strong features which distinguish the trends. The explanation for this is that webpages related to the trend contained words strongly related to the topic, i.e. for the hashtag "#helphaiti", the strongest features were "haiti", "earthquake", "donate", "disaster", and "relief". The Decision Stump, due to the single level tree, cannot reduce the number of false-positives using further decisions.

Figure 16 shows the F1-measure results of using the C4.5 decision tree classifier with 100 features on the webpage content. The number of webpages belonging to the trend being classified (the positive class) is shown on the x-axis with the F1-measure on the y-axis. Once again, the reason for the variance reducing and the increased F1-measure as the number of positive webpages linked, is that the percentage of suspended tweets which make up a trend as the trend size increases, decreases and this causes the classifier to be less likely to learn the suspended tweets as part of the trend.

We also look at the time (ms) taken to build the classification models and to classify individual webpages. Table 13 shows the average time to build the model (training time) and time to classify (test time) a single webpage using different classifiers and number of features. We see that the training time for naïve Bayes and Decision Stumps are the lowest, and C4.5 decision trees taking the longest time. In terms of classifying a webpage once a model is built, both C4.5 decision trees and Decision Stumps are the fastest, while the naïve Bayes is relatively slow. The time the naïve Bayes takes to classify a single webpage associated with a tweet increases by over 600% when increasing the number of features from 1000 to 5000 features.

5.4.3 Classification using Tweet Text and Webpage Content combined

After studying the performance of the classifiers on the individual sets of features, a natural next step is to combine the predictions from the classifiers of the tweets and associated webpages to a combined prediction. There are two main reasons for doing so:

1. Tweets might not contain enough text to make an accurate classification all the time. Some tweets might only contain a URL and a hashtag associating it with a trend. In

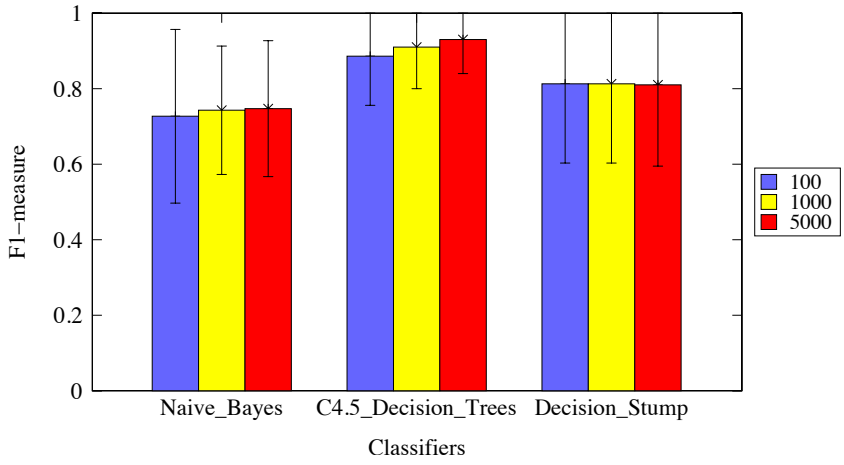


Figure 15: Results of classification on webpage content using different number of features. Error-bars represent one standard-deviation of the F1-measure.

Table 13: Average time in ms, to train and test the classifiers over the webpage content.

Webpage content	***Number of features***		
Training Time (ms)	*100*	*1000*	*5000*
Naïve Bayes	0.17	1.19	7.06
C4.5 Decision Tree	0.62	9.46	-
DecisionStump	0.17	1.25	7.53
Test Time (ms)	*100*	*1000*	*5000*
Naïve Bayes	0.17	0.90	5.96
C4.5 Decision Tree	0.05	0.11	-
DecisionStump	0.05	0.13	0.39

this case, classifying the webpage content associated with the tweet might help make an accurate classification on whether or not the tweet actually does belong to the trend.

2. It might be easy for a spammer to camouflage a tweet to look like it belongs to a trend, but it would take more effort (on the part of the spammer) to also camouflage the webpage linked in the tweet to look like it belonged to the trend. Even if the spammer were to do so, they would have to customize their webpage for each trend that they wish to spam, and this would likely be prohibitive.

We compare two methods of combining the classification results from tweets and from the associated webpages, outlining possible improvements to this method in Section 5.4.5. The 'AND' and 'OR' operator are commonly used to check if results from two statements

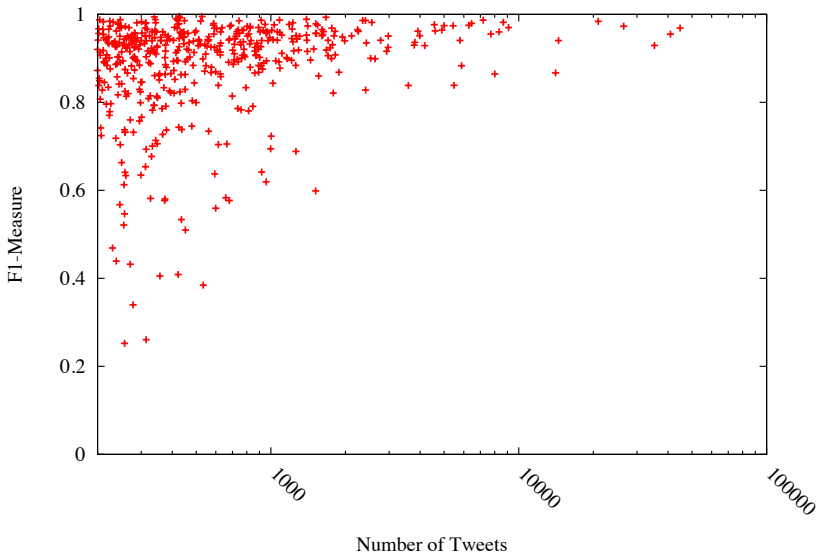


Figure 16: F-measure of C4.5 decision tree classifier using 100 webpage content features.

agree or disagree, and in the case of combining classification results, we will use these to check if the prediction over a tweet matches the prediction over the associated webpage content. In the case of the 'AND' operator, we will mark a tweet as belonging to a trend, if the prediction for the tweet 'AND' the prediction for the webpage associated with the tweet both predict that the tweet belongs to the trend. In the case of the 'OR' operator, we will mark a tweet as belonging to a trend, if either prediction for the tweet 'OR' prediction for the webpage associated with the tweet predict that the tweet belongs to the trend. Intuitively, by using the 'AND' operator, we ensure that both predictions on text and associated webpage agree to classify it in the trend, causing an increase in the number of false-negatives and true-negatives. On the other hand, the 'OR' operator, will cause an increase in the number of true-positives and false-positives.

Figure 17 and Figure 18 show the results based on combined classification using the 'OR' operator and 'AND' operator, respectively. The x-axis displays the classifier used for the tweet text, whereas each bar within a cluster denotes a classifier used to predict the label for the associated trend. All classifiers were trained and tested using 100 features. The y-axis shows the value of the F1-measure achieved by a combined tweet text and webpage content classifier.

We find that the combined classifier using the 'OR' operator performs better on average than the combined classifier using the 'AND' operator. This as previously mentioned in the intuition, is due to the 'OR' operator leading to more true-positives and perhaps even a few more false-positives, to the point of performing better than the best individual 100 feature classifier used. One of the possible reasons for not achieving results closer to 1, is likely due to trends which are largely made up of suspended tweets. We are currently investigating this further.

In the case of combining classifiers using the "AND" and "OR" operator, both classifiers would have to be trained on the trend, and at least one of them would have to be probed at the time of making the determination of whether the tweet and associated webpage belong to the trend. This is similar to the common practice of short-circuiting boolean expressions. In the case of an "AND" operator, a single negative prediction will result in the final label being negative, whereas in the case of the "OR" operator, a single positive prediction will result in the final label being positive. In respect to the tweets, the tweet text would be easier to get a classification label for, especially since obtaining a classification for the webpage would involve fetching the destination webpage followed by tokenizing and feature reduction of all the webpage content. Thus the minimum required time to classify a tweet using the combined approach is the time taken to classify the tweet text, plus an estimated $\frac{1}{2}$ of the time taken to classify the webpage content (not taking into account time to fetch, tokenize, and reduce features).

5.4.4 Manual Verification of a Subset of Results

In our manual vetting of the results we reviewed a few hundred tweets and associated webpages. We make a number of interesting observations based on what we found:

1. "Camouflage tweets". Such spammy tweets had legitimate looking tweet text, but the link pointed to a webpage which was unrelated to a trend. An example of this is the tweet "NFL Sunday Night Betting Preview - Brett Favre vs. Kurt Warner (maybe) [URL removed]" which although related to the trend "NFL", linked to webpage content for the dating site "Friend Jungle".

2. Many of the false-negatives were tweets properly classified as not belonging to the trend. The reason they were counted as false-negatives is that they belonged to Twitter accounts not yet suspended. For example the tweet "My Top 3 Weekly - Play Free Online Games [URL removed]" and associated webpage were both classified as not belonging to the "My Top 3 Weekly" trend, but the Twitter account associated with this tweet had not been suspended.

3. Some false-positives belonged to the trend they were tweeted in, but were still suspended. These were actually tweets and webpages to legitimate content related to a topic, but often with inappropriate or obscene language. An example is the tweet "Check this - Nicole Parker-Best of Britney Spears Part 2 [URL removed] Plz ReTweet! #musicmonday" related to the trend "#musicmonday", which links to parody video on Britney Spears. It is likely that the account posting this tweet got suspended for the sexually suggestive content of the parody.

5.4.5 Discussion

C4.5 decision trees perform the best and would likely be resilient to a certain amount of adversarial interference, but the time taken to build a model for a C4.5 decision tree is a hindering factor in using such classifiers. The Decision Stump classifier performs quite well and can be trained very quickly, although can be easily overcome by adversaries. Naïve Bayes although the most robust, takes the longest on classifying each tweet. An advantage of the naïve Bayes classifier is its suitability for online-learning, namely, being updated based on user feedback without a large re-training time. Therefore we believe that using naïve Bayes with a 100 features, or perhaps less depending on the maturity of the trend, would be ideal in classifying tweets and their associated webpages. We plan on studying the applicability of using varying feature set sizes, based on the number of tweets belonging to a trend and the effect of classifying recurring or multi-day trends.

Working with a 140 characters of text or less, has been both beneficial and challenging in that we find that for a particular trend, usually a small selection of features are adequate in being able to accurately capture tweets belonging to a trend. On the flip side, the challenge

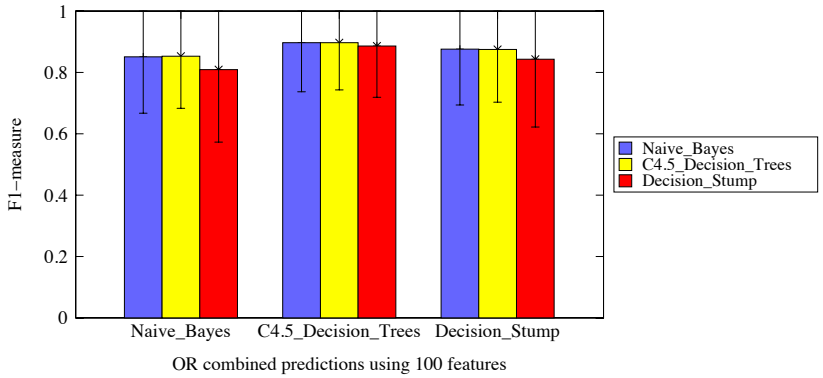


Figure 17: Results of F1-measure based on prediction from tweet text and webpage content classifier combined using the 'OR' operator.

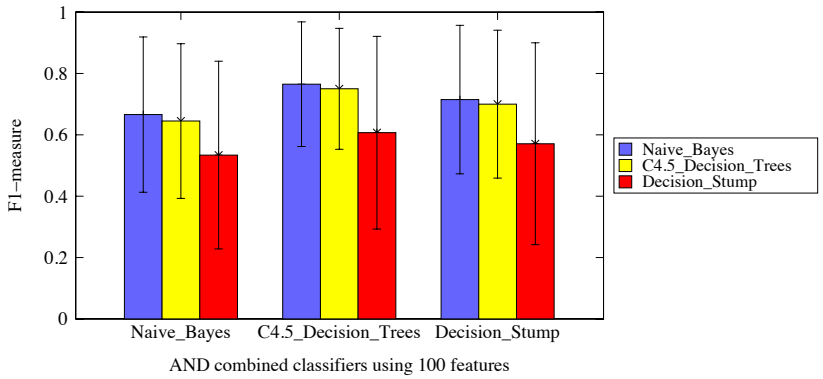


Figure 18: Results of F1-measure based on prediction from tweet text and webpage content classifier combined using the 'AND' operator.

is that as tweets contain pointers to allow gathering of further information, it is very easy for a spammer or a miscreant to camouflage the tweet—make the tweet text look legitimate while making the link point to a spammy webpage. It is due to this that we also crawl webpages associated with tweets and use such content in our classification.

5.5 Related Work

Twitter has largely been studied for its network characteristics and structure. The largest such measure to date has been done by Kwak et al. [61] who study over 106 million tweets and 41.7 million user profiles. They investigate network characteristics ranging from basic follower/following relationships to homophily (tendency for similar people to associate with one-another) to pagerank. They also study trends in relation to the similarity to topics on CNN and Google Trends, further characterizing most trends as being active. Huberman

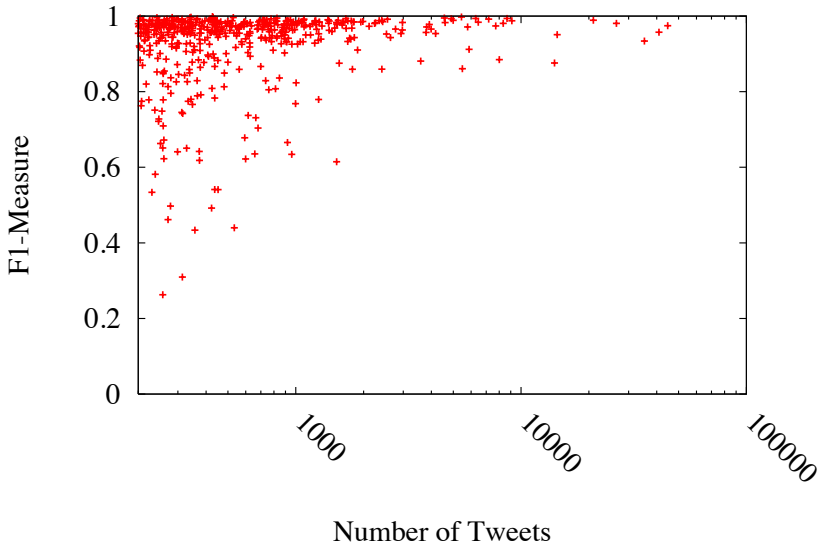


Figure 19: F-measure of combined classifier using naïve Bayes with 100 features on the tweet-text and C4.5 with 100 webpage content features.

et al. [45] and Krishnamurthy et al. [59] also study the characteristics of Twitter, with the latter identifying users, their behaviors as well as geographic growth patterns. Java et al. [53] did an earlier study on the Twitter network a year after its growth and used the network structure to categorize users into three main categories: information source; friends; and, information seeker. Information sources they note, were "found to be automated tools posting news and other useful information on Twitter".

In terms of spam detection on Twitter, Yardi et al. [105] looked at the life-cycle and evolution of a trend, with focus on spam. Although they do identify some behavioral characteristics that can be used to identify spammers, they do conceded that such behavioral patterns do come with their false-positives. We plan to investigate if we could combine such techniques to come up with a robust technique to keep spam messages out of trending topics. Other attempts at spam detection on Twitter include, Spamdetector [94], which is a tool to apply "some heuristics to detect spam accounts". Details on this project are unavailable, but their effort is continuing based on users' feedback to bots on Twitter.

Twitter themselves have taken a proactive approach to detecting spam on their network. In as early as July 2008 [89] and August 2008 [90], Twitter acknowledged an online battle with spammers and seemed to respond with placing limits on followers and delete spam

accounts by using the "number of users who blocked a user" as feedback. Further steps were taken in November 2009 [31] and in March 2010 [43], first in relation to reducing the amount of clutter posted to trends (although details on their approach are limited), and second in relation to introducing a URL filter for protecting against phishing scams.

There has been a lot of related work [86] on using text-based classification on e-mails to classify them as spam and non-spam. Such techniques could be directly applied to Twitter, but due to the short messages and camouflage to get users to click links in such posts, we believe that such techniques would have limited success. Our previous work [49] on classification of social profiles in MySpace is a precursor to this work, as that looks at zero-minute classification of social profiles using static profile information. Due to the limited information that Twitter gathers from a user during account creation, the previous work would likely need to be applied in conjunction with other techniques, such as the one being proposed in this chapter, to increase accuracy.

5.6 Conclusion

We research the non-trivial problem of text classification in a restricted environment of only 140 characters, such as that presented by trend stuffing on Twitter. Our approach to the problem included first modeling tweets that belong to a trend using text-classification on the text of the tweet itself. We followed this with building a meta-model for webpages linked in tweets that belong to a trend. Finally, we combined both the tweet text model and the webpage content model to increase the performance in classifying tweets that belong to trends. In each of the steps, we use additional information about suspended tweets to validate our results. To make text-classification on the tweet text and the webpage content feasible, we first used Information Gain to reduce the number of features from over 12,000 and over 500,000, to less than 1,000 features and less than 5,000 features respectively. For tweet text features and the webpage content features, this accounted for a reduction of over 91% and 99% respectively. We found that even with reductions this large, the Information Gain of additional features to be on the order of a thousandth of a bit.

C4.5 decision trees achieve the highest F1-measure on both the classification of tweet text

and associated webpage content, resulting in an F1-measure of 0.79 and 0.9 respectively. Naïve Bayes performs well on the tweet text with a result of 0.77 and not very well on the webpage content. On the other hand, Decision Stumps perform well on the webpage content with an F1-measure of 0.81 and not very well on the tweet text. We find by combining predictions from classifiers on the two sets of features, we perform well on most combinations of classifiers in the case of the 'OR' operator. In most cases, we do not find a significant difference between classification using 100 features and over a 1000 features. In addition we also compare the time taken to build a classification model and to test tweets or webpages against each classifier. We find that C4.5 decision trees take the longest to build a classification model, but are very quick at testing an instance. Naïve Bayes takes a much shorter time to build a classification model, but takes longer at testing each instance.

PREVENTING ABUSE OF ONLINE COMMUNITIES

PART II

Protecting against information leakage attacks

by

Danesh Irani

CHAPTER VI

USER'S SOCIAL FOOTPRINTS

Social networking sites are big. MySpace and Facebook both have over 250,000,000 accounts [14, 73] and are still growing at a rapid pace [28]. As a social network gets larger, they attract more users and share even more information.

Unfortunately, threats of private information leakage increase along with the growth of social networks. As a prerequisite for participating in most social networking sites, a user has to create a profile, enter some basic information, and is encouraged to enter detailed information relating to the purpose of the site. For example, Last.fm is a music streaming service, and it encourages users to enter details about their favorite artists. Due to the large number of social networking sites currently out there and the increasingly social nature of the Web, most users belong to more than one social networking site. These users assume that the information provided will be kept within the boundaries of the social networking site and that the privacy policies across sites are standard.

The danger of this implicit assumption is that many users do not realize how much information could be revealed by blurring the boundaries of these social networking sites. One danger in blurring these boundaries is that any information disclosed at one site could be combined with information at other social networking sites. We refer to the resulting combination of the information revealed by multiple social networking sites as a user's *online social footprint*.

We study the threats associated with an online social footprint by leveraging data collected from an online-identity management site. Online-identity management sites allow a user to provide links to all their social network profiles. We crawled one such site, retrieving over 13,990 profiles and 80,357 potential links to social network profiles. We use this data to preform quantitative measures with respect to the size of a person's online social footprint and the ability of an attacker to reconstruct such a footprint (without the aid of

an online-identity management site).

In this chapter, we make two main contributions:

- *Measure the size of a user's online social footprint* - We find that an active user has an average of 5.7 profiles, and 28% of those profiles profiles are in the top 15 social networking sites. Based on a sample of 9 personal information fields, we find that a person's online social footprint size increases from an average of 4.3 fields to 8.25 fields when the number of profiles in a person's online social footprint increases from 1 to 8 profiles.

- *Investigate how easy it is to reconstruct a user's online social footprint* - We show that an attack on targeted individuals is possible without having to use network based de-anonymization techniques [76,77]. First, assuming an attacker has prior knowledge of a single pseudonym, we measure the number of other social networking sites he can find. Our results show that an attacker would be able to find over 40% of a person's other social networking sites for most of the dataset in the best case (assuming the attacker knows the most-common pseudonym). Next, we assumed an attacker knows the person's name. In this case, the attacker will try and guess a person's pseudonym, and based on our data, he is able to find approximately 10-35% of a person's other social networking sites. Although this method might yield a few false-positives, we evaluate a technique which could be used to calculate the likelihood of two profiles belonging to the same user.

We expect that our findings will increase the awareness of the threat caused by large online social footprints and promote protection mechanisms against this threat.

6.1 Background

6.1.1 Online Social Footprints

A user's online social footprint is the online information that is available about him by aggregating his social networking profiles. Essentially, this footprint characterizes a user's social networking activities. To illustrate, Figure 20 shows a user named Bob Smith and his

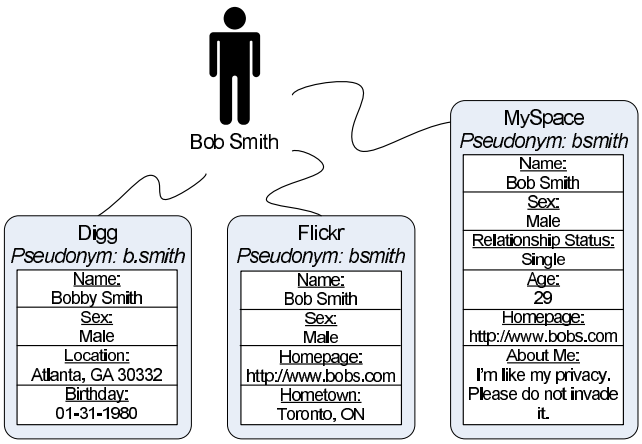


Figure 20: Online Social Footprint

online social footprint, which was constructed with information from three social networking sites (identified by two unique pseudonyms). Individually, each of these sites reveals between 4 and 6 pieces of information (e.g., age, sex, etc.); however, if the sites are linked together, 8 pieces of information are displayed about Bob. This combined view of Bob's information represents his online social footprint.

6.1.2 Threats of Information Leakage

Threats associated with Information Leakage have been discussed in previous research [41, 55], but we briefly describe them here for completeness.

- *Online stalking* - As users spend more time online and reveal more information about their activities, they become more susceptible to digital stalking. Using Twitter, stalkers can view "tweets" about their victims' current activities. On Last.Fm, stalkers can determine what music their victims are listening to. Utilizing Del.icio.us, stalkers can identify the surfing habits of their victims.

- *Compromising personal accounts* - Most login-based Web sites allow users to "recover" their passwords by answering some personal questions about themselves. Answers to common questions such as date of birth, address, or hometown are sometimes inadvertently made public on social networking sites. A recent example of this type of compromise occurred when Sarah Palin's e-mail account was hijacked because the

recovery question for her Yahoo account was discovered.

- *Customized spam/phishing* - Occurrences of spear spam/phishing have already been observed, and with an abundance of personal information, such techniques can be made more effective to the point where the user might be fooled into believing that the email is legitimate due to the amount of personalized information it contains.

6.2 Online Identities and Data Collection

Identity management sites allow users to manage their online identities by enabling them to provide links to their social networking sites. One can use an online identity to determine a user's social footprint by visiting each profile that is associated with the online identity and identifying the pieces of information that are revealed by each profile.

In August 2008, we crawled the publicly available online identities stored by one such identity management site. During this crawl, we collected 54,600 users' online identities, and of those identities, 13,990 were labeled active (i.e., the identity contained one or more links to a profile on a social networking site). From the 13,990 active identities, we found 80,357 links to social networking profiles. Then, we identified links pointing to the top 15 most common social networking sites, which accounted for 21,764 profile links. Next, we proceeded to crawl the profiles associated with each of these links, obtaining a total of 21,764 profiles. The distribution of the number of profiles crawled for each site is shown in Table 14.

The profiles and links were entered manually by the users of the identity management site. However, some of the links were not associated with the user entering the data. For example, a number of the links pointed to profiles belonging to the users' friends or celebrities. In a few extreme cases, the links formed a link farm that was used to promote other sites/profiles. To minimize the effect of these links and maintain an accurate representation of a person's online identity, we removed these unrelated links by implementing a few heuristics. For example, we removed any links that were duplicated across profiles because we had no way of knowing which profile was the legitimate owner of those links.

Table 14: Number of profile links.

Social Site:	***# of Profiles***
Blogspot	2625
Del.icio.us	1959
Digg	759
Facebook	1840
Flickr	3646
Last.Fm	1540
LinkedIn	1879
LiveJournal	645
MySpace	1677
Technorati	497
Tumblr	607
Twitter	1901
Wikipedia	280
Wordpress	926
YouTube	747

To investigate the amount of information leaked by this dataset, we wrote parsers for 10[1] of the top 15 social networking sites. The parsers performed a great deal of post-processing to merge fields with semantically similar meanings but syntactic differences. An example of this is "Age" and "Date of Birth", as Age can be represented as a coarse granularity date of birth.

Table 15 shows the amount of information parsed from each profile for a small subset of available fields. For each field, we represent the amount of information as a percentage of the number of profiles that displayed this value on a particular social networking site. If a '-' is present, it means the field was not revealed by the site in question.

6.3 Size of a user's online social footprint

From the data we collected, the number of users with profiles on multiple social networking sites follows a Zipfian distribution as shown in Figure 21. The number of users exponentially decreases with an increase in the number of profiles (e.g., we have 5,797 users with one profile link and 327 users with 10 profile links). We observe a similar trend with the top 15 social network profile links (e.g., we have 3,335 users with one top 15 profile link and 40 users

[1] A few sites were not parsed due to technical and legal challenges associated with parsing the site or the low value of information found on the site.

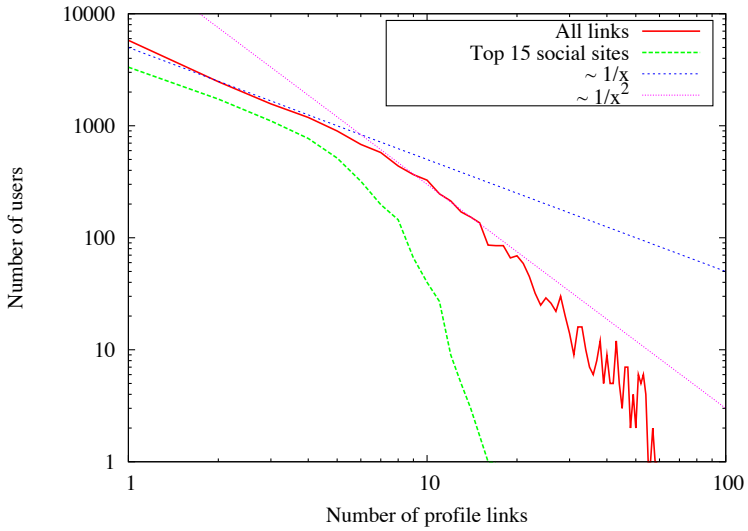


Figure 21: Number of users against size of social network footprint

with 10 top 15 profile links). On average, a user has 5.7 links to other sites, and an average of 1.6 of those links point to a social network in the top 15. Users with a link to the top 15 social profiles have an average of 2.6 profiles.

Looking at 9 basic fields, a subset of which are shown in Table 15, we measure the size of an online social footprint. For a user with one social networking site, the online social footprint size is 4.3 fields on-average, whereas the online social footprint size is over 8.25 fields, on-average, for a person with 8 or more social networking sites. Figure 22 shows the size of a user's online social footprint growing, with an increase in the number of social networking sites.

Depending on the social network being used, the information can vary widely. For example: Delicious tracks a user's favorite URLs; Flickr stores user's pictures, and Last.Fm tracks the music a user is listening to. We see that sites like Facebook, Flickr, and YouTube collect and display more basic personal information than sites such as Delicious and Twitter.

Table 15: Subset of fields collected for each social networking.

Social Site:	***Name***	***Location***	***Sex***	***Relationship***	***Hometown***	***Homepage***	***Birthday***
Del.icio.us	-	-	-	-	53	-	-
Digg	100	67	55	-	-	-	30
Flickr	73	58	82	59	51	74	-
Last.Fm	82	-	87	-	76	77	-
LinkedIn	100	88	-	-	-	-	-
LiveJournal	93	69	-	-	-	68	64
MySpace	94	98	100	72	40	-	100
Technorati	94	-	-	-	-	-	-
Twitter	100	93	-	-	-	89	-
YouTube	68	-	-	-	29	57	73

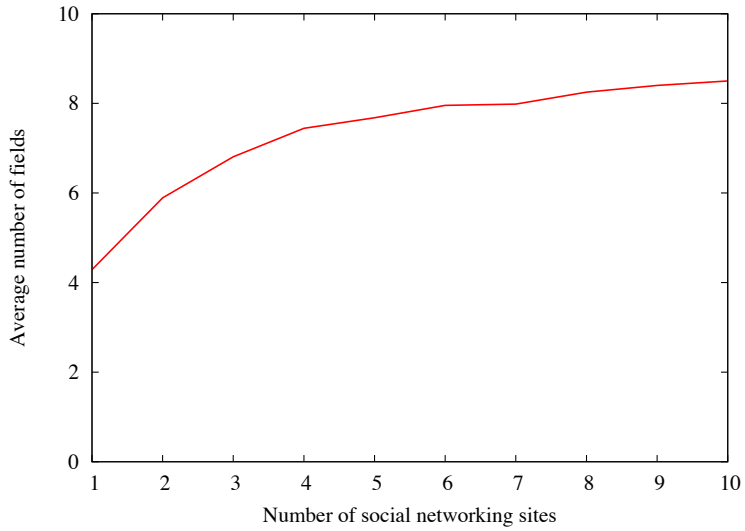


Figure 22: Average size of an online social footprint increasing with number of social network sites

6.4 Reconstructing a person's online social footprint

Each social networking site reveals a limited, sometimes unique, amount of information about a given user, but if an attacker coalesced the information from multiple social networking profiles for that user, the amount of disclosed information would increase significantly. In this section, we explore the feasibility of profile aggregation techniques that can be performed by attackers.

Previous work on aggregation techniques (or de-anonymization techniques) [76, 77] have focused primarily on network based matching techniques. Unfortunately, these approaches typically require knowledge of a large portion of a social network graph with overlapping sections, which is very computationally expensive and often impossible to obtain. Thus, to avoid these issues with previous techniques, we propose using pseudonyms to match profiles across multiple social networking sites.

6.4.1 Prior knowledge of pseudonyms

Our first aggregation technique assumes that an attacker knows one of a user's pseudonyms from an e-mail address or another source. Using a known pseudonym, we investigate what percentage of a user's social networking sites can be found by an attacker. In the best-case (from an attacker's standpoint), the known pseudonym is the most frequently used by a

user, and as a result, it allows the attacker to access a maximum number of the user's social networking sites. On the other hand, the known pseudonym could be the least frequently used by a user, revealing a minimum number of the user's sites (this represents the worst-case).

Figure 23 illustrates the success rate of this approach by an attacker. The x-axis shows the number of unique id's for a user, and the y-axis shows the percentage of the user's total number of sites that an attacker can find. For users with two unique pseudonyms, the figure shows that an attacker can obtain 62% of their social networking sites in the best-case (i.e., with knowledge of a user's most frequently used pseudonym) and 38% of their sites in the worst-case (i.e., with knowledge of a user's least frequently used pseudonym). Over 98% of the users in our dataset have 4 or less unique pseudonyms. Thus, an important observation from the figure is that an attacker can find more than 40% of those users' social networking profiles in the best-case. Even in the worst-case, an attacker can still find over 17% of those profiles.

Although the figure only illustrates the situation in which an attacker knows a single pseudonym, we also investigated scenarios in which an attacker knows two or more of a user's pseudonyms. As expected, the success rate for these attacks increases dramatically. With knowledge of only two of a user's pseudonyms, an attacker will find over 60% of the user's social networking profiles in the best-case (and more than 35% of the profiles in the worst-case).

6.4.2 Inferring a pseudonym from a name

Another method of aggregating a user's profiles involves guessing that user's pseudonyms based on real name information. This method has not been sufficiently explored in previous literature, and in this section, we investigate the technique's ability to match a user's profiles.

To perform the aggregation process, we use three categories of inference rules. To achieve a baseline result, the inference rules are intentionally simplistic, and each category is limited to approximately 30 rules. The three categories are as follows:

1. *First name and last name* - includes guesses of a pseudonym that include the first

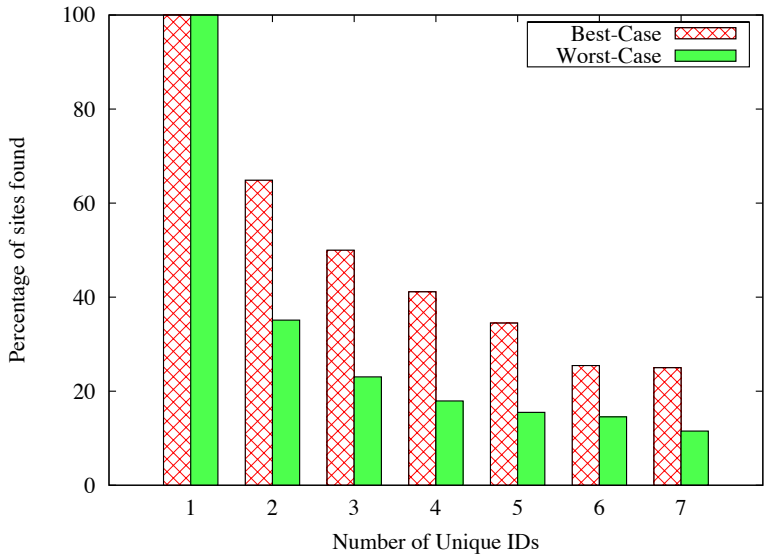


Figure 23: Number of sites identified with a single pseudonym in the best-case and worst-case.

name and last name. This category also includes tests that incorporate a subset of the first name and/or last name. Examples of guesses include "{FirstName}{Separation-Character}{LastName}", "{FirstName}{LastInitial}", and "{FirstNameSubstr(3)}{LastName}" (where Separation-Character is one of [-,;_.+,]).

2. *First name* - includes guesses of a pseudonym that include the first name or a substring of the first name. Examples of guesses include "{FirstName}", "{FirstNameSubstr(3)}", "{FirstName}007/69", etc.

3. *Last name* - includes guesses of a pseudonym that include the last name or a substring of the last name. Examples of guesses include "{LastName}", "{LastNameSubstr(3)}", "{LastName}007/69", etc.

To illustrate the power of pseudonym guessing, we begin by guessing a user's pseudonym in each of the top 15 social networking sites. Figure 24 shows a stacked bar graph that illustrates the number of pseudonym matches found for each site using each of the inference rule categories. Each bar consists of three sub-components (one sub-component for each category). The figure shows that LinkedIn has the highest number of matches for our guess-based approach, presumably due to the professional nature of the site. Blog web sites such

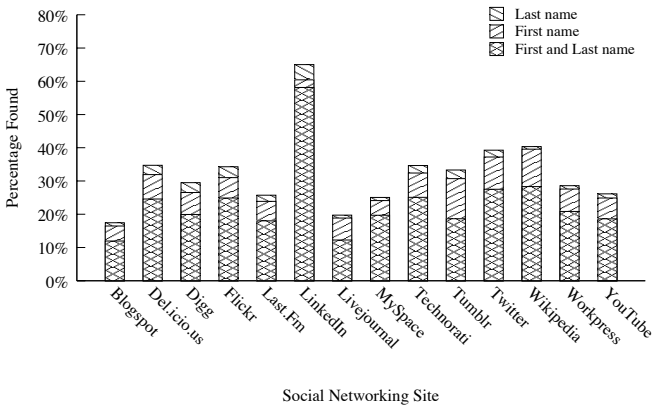


Figure 24: Percentage of pseudonyms found by inference rule category for each social network site.

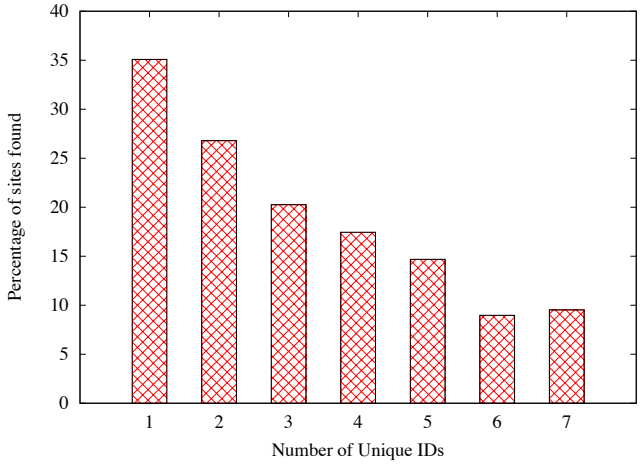


Figure 25: Number of sites identified per pseudonym using names as guesses.

as Blogspot, LiveJournal, and Wordpress have a lower than average matching percentage. One hypothesis for this result is that most people name their blogs about the topic they plan to write about (i.e., they do not use their real name to title their blogs).

Another important observation from Figure 24 is that a large percentage of the matches appear in the "First and last name" category. In fact, over 50% of the overall matches resulted from one of this category's inference rules: {FirstName}{LastName}. Additionally, 14% of these matches were found on LinkedIn.

Using only the simple guesses described above, we also investigated the percentage of a user's profiles that can be found using guess-based aggregation. Figure 25 shows the percentage of a user's profiles that were found using this guessing approach. From the

figure, we see that even with limited guesses, an attacker can find up to 35% of a user's social networking sites (if a user only has one unique pseudonym). Additionally, for users with six or seven unique pseudonyms, attackers are still about to find around 10% of their sites.

Although we used relatively simple matching criteria, two profiles with identical pseudonyms might not belong to the same user (i.e., we might have a false positive). This problem is exacerbated if one were to guess pseudonyms more aggressively. As the aggressiveness of the guesses increases, the number of false positives increases as well. To truly estimate the false positive rate for our guesses, we would need to check if the pseudonym existed on the respective social networking sites for every one of the generated guesses. This would be expensive both computationally and in terms of network bandwidth. To overcome these limitations, we introduce a method in the next section that allows us to gain confidence about whether or not two profiles are the same by comparing information within the profiles.

6.4.3 Matching profile information

In this section, we offer an approach for determining if two profiles belong to the same user. This technique can be used in multiple ways including matching profiles together in the case when there are no pseudonyms (or pseudonyms are hidden, as is the case with Facebook) as well as eliminating false-positives.

To check if two profiles belong to the same user we need to check if fields common to both profiles are equal or similar in value. In this section, we focus our attention on categorical or single text fields (e.g., "First name", "Hometown", etc.) and leave investigating similarity across free text entry fields (e.g., "About me", "Interests", etc.) as future work.

In our dataset, for profiles that have duplicate fields, we investigate the consistency of information entered. The consistency of a field is calculated by taking all possible pairwise combinations ($\binom{n}{2}$, where n is the number of values a field takes on within a profile and across the dataset respectively) and checking if the values are equal.

Figure 26 shows the amount of consistency between fields within a profile and across the dataset. We performed this evaluation to determine how well two matching fields can

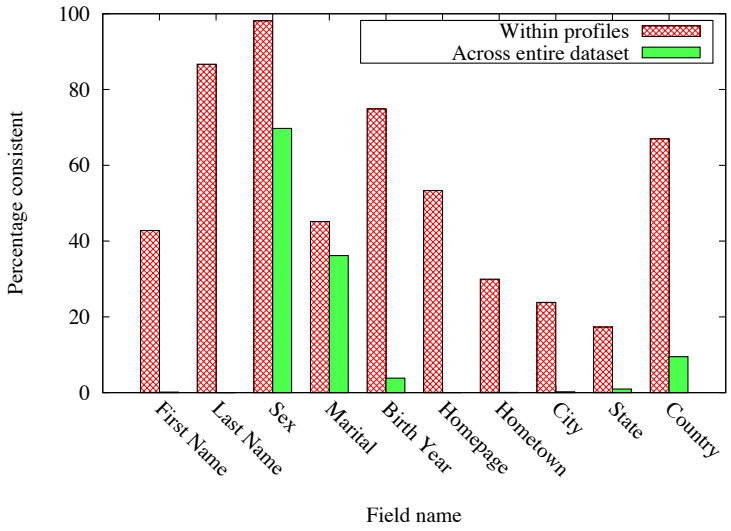


Figure 26: Consistency of fields within a profile and across the entire dataset.

predict two matching profiles. From the figure, we observe that "Sex" is the field with the highest consistency across profiles; however, this field is not a strong signal because the consistency of "Sex" against the dataset is very high. As a result, "Sex" is not a very discriminatory field. Last name, birth year and country all seem to be strong signals with higher than average matching ability within profiles and low matching ability across the dataset. Therefore, any of these fields should be a good candidate for checking if two profiles belong to the same user.

The consistency of certain fields like Martial status and Country could be improved by cleaning up the data and assigning synonyms to certain values. For example, one social network site might not distinguish between "Marriage" and "Common-law" whereas another site might allow the user the option to do so. The value in this case is not inconsistent; it is simply semantically different. Similarly, more complex location-based fields such as Street, State or City could be improved by taking into account different ways of representing the same value, (e.g., a state Georgia could be represented in one social network as Georgia and as GA in another). Ideally, one would simply be able to feed two locations into a mapping service and find the distance between the two locations, applying a radius threshold to the value to determine if they are in the same locality.

Instead of assigning a simple binary value (i.e., consistent or not) to particular field

matches, a more flexible approach involves assigning a consistency probability based on a measure of "how far" one value is from another. For numeric fields, a simple measure of how close the two numbers are divided by the standard deviation can be used as a measure of how similar two fields are. For textual fields, string similarity metrics (e.g., Cosine similarity, QGramsDistance, etc.) can be applied.

To illustrate the value of this approach, consider the "Birth Year" field. It is consistent 75% of the time within profiles and only consistent 4% of the times across the dataset. As a result, this field is a good choice for matching between profiles. Given two profiles that possess the "Birth Year" field, if those two values match, it means that (with a certain probability) the two profiles belong to the same user. After investigating the distance measure (in the numeric case, the absolute value of the subtraction), we found that within profiles the sample standard deviation (including Bessel's correction) is an average of 1.8 years. However, across the dataset the sample standard deviation is 12.7 years. In the event that two "Birth Year" values do not match, this allows us to say with some probability what the likelihood is that the profiles belong to the same user (if the difference between values is under 1.8 years, the likelihood will be much higher than if it is closer to or over 12.7 years).

Even after performing the aforementioned correlation analysis, it might not be possible to easily match profiles. Different social networking sites might not reveal the same pieces of information, or they might not reveal enough of the same pieces of information to be able to make a strong match. To help alleviate this situation, Table 15 shows the amount of information that is available for the most popular sites. This information can be used to determine which sites are most susceptible to this type of matching, and it also provides an optimal match order.

6.5 Related Work

A brief outline of areas in social networking sites which can compromise a user's privacy are discussed in Chew [27]. Some of the areas she discusses include "Activity Streams", "Unwelcome Linkage", and "Merging Social Graphs". Further, Gross and Acquisti [41] highlight potential attacks on privacy and also show that a minimal percentage change

default lax privacy settings.

Krishnamurthy [60] present statistics on privacy settings of two popular Social Networks. They show that 79% of MySpace allowed their profile, friends, comments and user content to be viewable. They also showed that a majority of users do not change default privacy settings of Facebook allowing anyone in the same "Regional Network" to view their profile.

Narayanan and Shmatikov [77] discuss techniques to de-anonymize large social networks using only network structure. The work presents another method by which relationships between profiles of the same user on multiple social networks can be identified.

6.6 Conclusion

As social networks continue to grow in size and importance, they begin to pose a number of interesting privacy challenges for their users. In this paper, we investigated one of those challenges: large online social footprints. Specifically, we have shown that targeted attacks on individuals are possible using techniques based on social networking pseudonyms. Based on our experiments, we have shown that an attacker can reconstruct over 40%, in the best-case from his perspective, of an individual's social footprint by using a single pseudonym. Additionally, an attacker can reconstruct 10% to 35% of an individual's social footprint by using the person's name.

The outlook we present in this chapter makes hope seem bleak, especially with the aim of social networks being to promote creating online-identities and sharing information between friends. Normal anonymization methods of k-anonymity and anonymized network leakage are not used when routinely displaying information to a potential friend, a curious wanderer, or a malicious attacker.

CHAPTER VII

MODELING INFORMATION LEAKAGE

Social networking sites are of great interest. They are, by nature, platforms for users to share personal information and experiences with friends and to find potential friends with common interests. Over one-third of all American adult-internet users use social networking sites [65], of which 51% participate in multiple social networks. To participate in a social network, a user has to create a profile to describe herself, and since accurate profile information allows more efficient networking, users have an incentive to fill in real information into their profiles.

Social networks usually contain different pieces of information about personal aspects of a user, as they target users with a particular interest (e.g., Flickr for photo sharing, Last.fm for music lovers). Users are encouraged to share information about these interests and in providing this information to a social network, most users assume that their profile information will be kept within the boundaries of the social network site. Boundaries between social network sites can be quite easily blurred using a number of techniques including deanonymization of many users using friend network graphs has been shown to be successful [77], and even targeted deanonymization of a user is possible [48].

Unfortunately, social network sites do not protect users from attackers who wish to combine the disparate pieces of information about a user from multiple sites. This often exposes more of the victim's information than from a single site. We call this problem "unintended personal information leakage" and define a *social footprint* to be the total amount of personal information that can be gathered about an online identity by aggregating available social networks.

This chapter presents the first study on the unintended personal information leak problem in multiple social networks with the aim of providing a model to measure it and proposing a cloaking solution based on our model. We demonstrate the usefulness of these models

and proposed solution on over 8,200 real-world social footprints.

More concretely, the first contribution of this chapter is towards a rigorous understanding of the unintended personal information leakage problem in social networks. We do this in two parts. First, we define measures to quantify the amount of information in a person's social footprint. These measures provide an upper bound, a lower bound, an information theoretic bound, and a probabilistic bound, on the amount of personal information in a user's social footprint. The latter two measures are required in order to take into account inconsistencies between personal information on multiple social networks. Second, using approximately 8,200 users' social footprints, we conduct an empirical study to evaluate the effectiveness of two attacks using a user's social footprint: an identification attack and a password recovery attack. We find that the number of users susceptible to the identification attack increases from 2% for one social network to 18% when using a person's social footprint. The number of users susceptible to the password recovery attack follows a similar trend. The second contribution of the chapter is using cloaking to reduce the precision or resolution of information released by a person's profile. Using "Birthdate" as a sample attribute, we show a reduction of 66% for the information theoretic and probabilistic attribute leakage measures when cloaking is applied (with a cloaking range of 5).

7.1 Attacks using Unintended Personal Information Leak

To analyze the threat of attacks using personal information aggregated from multiple social network sites, we need to define a set of attributes required for each attack. We investigate two attacks previously demonstrated using manual techniques or other scenarios. The sets of attributes defined for them are:

Physical Identification Attack: Personal Identification Information (PII) is "information which can be used to distinguish or trace an individual's identity" [54]. Most social networks do not explicitly reveal PII, although many do reveal a person's birthdate, gender, or location. Previous research by Sweeney [95] on U.S. Census data showed that this particular set of attributes (where location is the City, Town, or Municipality in which the person resides) can uniquely identify 53% of the U.S. population. Therefore to measure the

amount of information required for a physical identification attack, we use the following set of attributes {Birthdate, Gender, Location}.

Password Recovery Attack: Many web sites allow legitimate users to recover their passwords by providing personal "secret" information (e.g., birthdate and address). The authentication factor of "something you know" relies on questions being personal in nature. Unfortunately, attackers often infer the answers from other sources and use this recovery mechanism to compromise accounts. For users that have publicly available social profiles, their social footprints provide a rich source of information for attackers to mine information about that user.

The exact information needed to recover passwords varies among sites and is sometimes also based on the type of questions a user selects. Based on common password recovery questions discussed in previous research [56,84,85], we define the set of attributes for a password recovery attack to consist of {Name, Email, Nickname, Location, Gender, Hometown, Homepage, Birthdate}.

7.2 Online Social Footprints

We have discussed social footprints in the previous chapter. Here we briefly review how the boundaries of social networking sites can be blurred, followed by a formal definition of a user's social footprint.

7.2.1 Blurring the Boundaries between Social Networks

Constructing a user's online footprint requires access to information about the same user on different social networks. This requires identifying a user's profiles across the boundaries of distinct social networks. Occasionally users use Online Identity Management sites to publish links to their profiles for different networks, which make the connections between social networks explicit. When users do not provide explicit links to their other profiles, targeted deanonymization of a user is possible [18, 48] and also deanonymization of a large number of users across social networks has been shown to be successful [77].

7.2.2 Definition of Social Footprint

An attribute f is a piece of information belonging to a person's profile. It is defined as a tuple $\langle f_a, f_v \rangle$ containing the value f_v, of an attribute named f_a.

A user's profile τ_s^u is a set of attributes, representing information obtained about a user u on social network site s. Using Figure 20 as a running example, Bob's Digg profile can be represented as $\tau_{Digg}^{Bob} = \{\langle$"Name", "Bobby Smith"$\rangle$, $\langle$"Sex", "Male"$\rangle$, $\langle$"Location", "30332"$\rangle$, $\langle$"Birthdate", "/1980"$\rangle\}$.

A user's online identity T is a list of profiles τ, which represent a user's online identity. In Figure 20, Bob's online identity has three profiles: τ_{Digg}^{Bob}, τ_{Flickr}^{Bob}, and $\tau_{MySpace}^{Bob}$.

Finally, a user's social footprint P^u is given by a union of the profiles in their online identity:

$$P^u = \bigcup_{i \in T} \tau_i^u$$

A subset of Bob's social footprint can be expressed as: $P^{Bob} = \{\langle$"Name", "Bobby Smith"$\rangle$, $\langle$"Name", "Bob Smith"$\rangle$, $\langle$"Name", "Bob Smith"$\rangle$, $\langle$"Sex", "Male"$\rangle$, ...$\}$.

7.3 Quantitative Definitions of Aggregate Attribute Leakage

7.3.1 Definition of Attribute Leakage

Attribute Leakage is a measure of the information that can be discovered about a particular attribute, given a person's social footprint. We define four measures for attribute leakage:

7.3.1.1 Attribute Leakage Upper-bound

Our first definition of attribute leakage simply checks for the presence of an attribute in a person's social footprint. It does not take into account consistency across attribute values from different profiles and hence is used as an upper-bound for the attribute leakage measure.

Definition $\phi_U(f_a, P)$—Attribute Leakage Upper-bound. Given an attribute name f_a and a user's social footprint P, the attribute leakage upper-bound is defined as:

$$\phi_U(f_a, P) = \begin{cases} 0 & \text{if } f_a \notin P, \\ 1 & \text{if } f_a \in P. \end{cases}$$

where $f_a \in P$ checks if an attribute tuple, with f_a as an attribute name, exists in P. The "$\in$" notation is used in this manner henceforth.

This measure is also useful in situations where an average over a group of users is calculated, as it can be interpreted as a count for the number of users. For example, if the average attribute leakage upper-bound is 0.66 for a group of users, that result can be interpreted as two-thirds of the group revealing the attribute.

7.3.1.2 Probabilistic Attribute Leakage

Our second definition takes into account the amount of consistency between values of an attribute across profiles. In this case, we use the relation of consistency being inversely proportional to inconsistency. Based on this, a simple measure could be inversely proportional to the number of unique values of an attribute, but this discards useful information on the probability of each value.

As a measure of inconsistency in the second definition, we use the probability of each unique value (of an attribute) to calculate the expected number of guesses an attacker would have to make before getting the right value.

Definition $\phi_C(f_a, P)$—Probabilistic Attribute Leakage. Given an attribute name f_a and a user's social footprint P, the probabilistic attribute leakage is defined as:

$$\phi_C(f_a, P) = \begin{cases} 0 & \text{if } f_a \notin P, \\ \dfrac{1}{\sum_{i=1}^{|U(f_a,P)|} i \times \rho(i)} & \text{if } f_a \in P. \end{cases}$$

where $U(f_a, P)$ is a set containing unique attribute tuples (with the attribute name f_a) in a user's social footprint P. $\rho(i)$ represents the probability of the i^{th} unique attribute tuple being picked from the other attribute tuples in P with the same f_a.

This measure matches the informal definition of attribute leakage. When all the attribute values are consistent, there is only one unique value, and the associated probability of the unique value is 1. When the attribute values are inconsistent, the number of unique

values increases and causes the sum in the denominator (or number of guesses) to increase. Thus, as the probabilistic attribute leakage increases, an attacker trying to guess the attribute's exact value would expect to make fewer guesses.

As an optimization, to obtain the minimum number of expected guesses, we order the list of unique attribute values with the most probable value to least probable value.

Using Bob's identity in Figure 20 as an example, we calculate the ϕ_C. As per our definition above, $U(\text{“Name”}, P^{Bob}) = \{\langle\text{“Name”},\text{“Bob Smith”}\rangle, \langle\text{“Name”},\text{“Bobby Smith”}\rangle\}$ and the probabilities associated with the two attributes are $\frac{2}{3}$ and $\frac{1}{3}$.

$$\phi_C(\text{“Name”}, P^{Bob}) = \frac{1}{\sum_{i=1}^{|U(\text{“}Name\text{”},P^{Bob})|} i \times \rho(i)}$$
$$= \frac{1}{1 \times \rho(1) + 2 \times \rho(2)} = \frac{1}{\frac{4}{3}} = 0.75$$

We note that in some cases, the values revealed for an attribute might not be factual in terms of the physical person it represents (i.e., a person might not be truthful in filling out their online profiles). We argue that this occurs in the minority of cases as it reduces the utility of a user's online profiles.

7.3.1.3 Information Theoretic Attribute Leakage

We use the same relation of a measure of consistency being inversely proportional to inconsistency. In the third definition, we use entropy [87], a measure of the amount of uncertainty associated with a variable, as a measure of inconsistency.

Definition $\phi_I(f_a, P)$—Information Theoretic Attribute Leakage. Given an attribute name f_a and a user's social footprint P, the information theoretic attribute leakage is defined as:

$$\phi_I(f_a, P) = \begin{cases} 0 & \text{if } f_a \notin P, \\ \dfrac{1}{1 + H(f_a, P)} & \text{if } f_a \in P. \end{cases}$$

where $H(f_a, P) = \sum_{x \in U(f_a,P)} p(x) \times \log_2 p(x)$. $U(f_a, P)$ is a set containing unique attribute tuples (with the attribute name f_a) in a user's social footprint P. $p(x)$ represents the probability of the attribute tuple x being picked from other attribute tuples in P with the same f_a.

Intuitively, entropy can be understood as the number of "smart" guesses it will take on average to find a value [22]. "Smart" guesses are assumed to be able to eliminate half the set of possible values on every guess, akin to using a binary search to find a number. We add one to the value of entropy to represent the fact that even after a binary search and knowing with certainty that the unguessed attribute value has to be true, we still need to guess/submit that value. Thus we can see the information theoretic attribute leakage measure as a representation of how many "smart" guesses it would take to correctly guess a value.

Using Bob's identity in Figure 20 as an example, we calculate the information theoretic attribute leakage. To calculate ϕ_I, we first need to calculate the entropy:

$$\begin{aligned} H(\text{“Name”}, P^{Bob}) &= p(\text{“Bob Smith”}) \times -\log_2 p(\text{“Bob Smith”}) \\ &\quad + p(\text{“Bobby Smith”}) \times -\log_2 p(\text{“Bobby Smith”}) \\ &= \frac{2}{3} \times -\log_2 \frac{2}{3} + \frac{1}{3} \times -\log_2 \frac{1}{3} = 0.92 \end{aligned}$$

Then, we use the entropy in the equation for attribute leakage:

$$\begin{aligned} \phi_I(\text{“Name”}, P^{Bob}) &= \frac{1}{1 + H(\text{“Name”}, P^{Bob})} \\ &= \frac{1}{1 + 0.92} = 0.52 \end{aligned}$$

Similar to the probabilistic attribute leakage, as the information theoretic attribute leakage increases, an attacker trying to guess the attribute's true value using "smart" questions would expect to make fewer guesses.

Comparing ϕ_C("Name",P^{Bob}) with ϕ_I("Name",P^{Bob}), we see that the value for ϕ_C is higher. Initially, this might not make sense because when using "smart" guesses, one would

expect to be able to guess the right value in fewer attempts. The explanation for this is, for a small number of unique values, making regular guesses can be faster than making "smart" guesses, akin to a linear search being quicker than a binary search under the same conditions.

7.3.1.4 Attribute Leakage Lower-bound

Our final definition of attribute leakage is a measure of consistent attribute values, which can be treated as a lower-bound for our attribute leakage measures. This is the strictest measure of attribute leakage where we require the value of an attribute to be leaked and all values to be consistent.

Definition $\phi_L(f_a, P)$—Attribute Leakage Lower-bound. Given an attribute name f_a and a user's social footprint P, the attribute leakage lower-bound is defined as:

$$\phi_L(f_a, P) = \begin{cases} 1 & \text{if } f_a \in P \text{ and } \forall i, j, \text{if } f_{ai}{=}f_{aj}, \text{ then } f_{vi}{=}f_{vj}, \\ 0 & \text{otherwise.} \end{cases}$$

7.3.2 Definition of Aggregate Attribute Leakage

Measuring attribute leakage for a particular attribute can reveal useful information, but to quantify the threat of a particular attack, we measure the amount of attribute leakage for the set of attributes required to execute the attack. We measure this in two ways: first, measure the total aggregate amount of attribute leakage for the set of attributes required for an attack; second, measure the total aggregate amount of attribute leakage only if all attributes are non-zero.

The advantage of the first method is that as the number of attributes (from a set of attributes defined for a particular aggregate attribute leakage measure) goes up, the amount of aggregate attribute leakage will go up. Thus, this method will approximate how close a user is to leaking all the attributes required for an attack. The second method has more intuitive meaning when looking at a group of users and trying to determine the amount of aggregate attribute leakage in terms of consistency and ease of attack.

As different attacks have a different number of attributes required for an attack, the total aggregate attribute leakage would vary depending on the number of attributes required. For instance, if two of the three attributes used for an attack are present, two-thirds of the attributes required are present; if two of the six attributes required for another attack are present, only one-third of the attributes required are present, but the total aggregate attribute leakage would be 2 in both cases. Conversely, the total is useful in emphasizing that more attributes are being leaked if two-thirds of the attributes required for the second attack are present, namely four attributes are being leaked. For purposes of comparing the aggregate attribute leak needed for various attacks, we normalize the aggregate measures to a 0 to 1 range.

We provide two definitions for normalized aggregate attribute leakage, based on the two methods described above.

7.3.2.1 Aggregate Normalized Attribute Leakage

Aggregate attribute leakage is the sum of attribute leakage for a set of attributes belonging to a person's social footprint. To normalize it to between 0 to 1, we divide by the number of attributes.

Definition $\Psi_X(F_a, P)$—Aggregate Normalized Attribute Leakage. Given a set of attribute names F_a and a user's social footprint P, the normalized aggregate attribute leakage is defined as:

$$\Psi_X(F_a, P) = \frac{\sum_{f_a \in F_a} \phi_X(f_a, P)}{|F_a|}$$

where X represents one of the definitions of attribute leakage—U, I, C, or L for Upperbound, Information Theoretic, Probabilistic, or Lowerbound, respectively.

7.3.2.2 Actionable Aggregate Normalized Attribute Leakage

Actionable Aggregate Attribute Leakage is the sum of attribute leakage for a set of attributes when all the attributes are leaked (in relation to a person's social footprint). Namely, the

actionable aggregate attribute leakage is zero if even one of the attributes is not-leaked or zero. We call this "Actionable Aggregate Attribute leakage" as for an attacker to use a set of attributes in a particular attack, they would need to have at least some information on all the attributes. To normalize this value to between 0 to 1, we divide by the number of attributes.

Definition $\Theta_X(F_a, P)$—Actionable Aggregate Normalized Leakage. Given a set of attribute names F_a and a user's social footprint P, the actionable aggregate normalized attribute leakage is defined as:

$$\Theta_X(F_a, P) = \begin{cases} 0 & \text{if } \exists f_a \in F_a \text{ and } f_a \notin P, \\ \Psi_X(F_a, P) & \text{otherwise.} \end{cases}$$

where X represents one of the definitions of attribute leakage—U, I, C, or L for Upperbound, Information Theoretic, Probabilistic, or Lowerbound, respectively.

Once again, the Upperbound Actionable Aggregate leakage is useful when an average over a group of users is calculated. This is because, for actionable aggregate leakage, it can be interpreted as the number of users which have released all the attributes required for a particular attack.

7.4 Online Identities and Data Collection

Identity management sites allow users to manage their online identities by enabling them to provide links to their social networking sites. Examples of such online identity management sites include ClaimId, FindMeOn, FreeYourID, and MyOpenID.

We crawled the public online identities stored by one such identity management site. During this crawl, we collected 8,268 active online identities—active online identities contained one or more links to a profile on a social networking site. We then identified links pointing to the top 15 most common social networking sites, which accounted for 21,764 profile links. We proceeded to crawl the profiles associated with each of these links, obtaining a total of 21,764 profiles, belonging to 8,268 users. Further details of the dataset can be found in the previous chapter.

To enable us to perform a detailed study on attribute leakages, we wrote parsers for 10 of the top 15 social networking sites—a few social networks were not parsed due to technical and legal challenges associated with parsing their sites. The parsers performed a great deal of post-processing to merge attributes with semantically similar meanings but syntactic differences. An example of this is "Age" and "Birthdate", as Age can be represented as a coarse granularity birthdate. We also standardized the value of certain attributes across social networks to semantically equivalent values. For example, if a MySpace user chose "In a Relationship", "Engaged", or "Married", we standardized the value to "In a Relationship".

Table 15 shows the amount of information leaked from each site for a small subset of available attributes. For each attribute, we represent the amount of information as a percentage of the number of profiles that publicly displayed this value on a particular social networking site. If a '-' is present, it means the attribute was not revealed by the site in question.

7.5 Experimental Evaluation and Results

Using the 8,268 social footprints collected, we begin by using the attribute leakage measures to quantify the amount of information that is discoverable for certain attributes. We then use the aggregate attribute leakage measures to quantify the amount of aggregate attribute leakage, with respect to the set of attributes required for attacks (described in Section 7.1).

7.5.1 Attribute Leakage Measures

We expect the overall attribute leakage of a particular attribute to be related to the number of social networks that reveal that attribute. To test this hypothesis and investigate the amount of inconsistency that exists between the attribute values revealed on social networks, we analyze the different attribute leakage measures for a subset of attributes. To compute these measures, we calculate the attribute leakage of each attribute in a user's social footprint, and then, we average the results across the entire dataset. The result of these calculations for a subset of attributes is shown in Figure 27. The y-axis shows the attribute leakage amount, and the x-axis shows the different attributes. For Figures 27-29, the Upperbound, Information Theoretic, Probabilistic, and Lowerbound labels represent

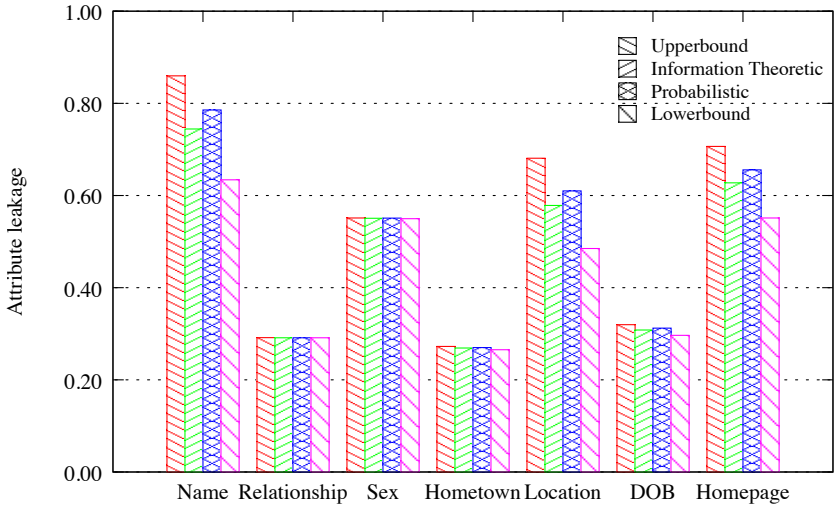


Figure 27: Attribute Leakages ($\phi_i(f_a, P)$) for a sample of attributes. Each attribute (f_a) is represented by a set of clustered bars on the x-axis and an individual bar represents the attribute leakages (ϕ_i) as defined in Section 7.3.1.

the ϕ_U, ϕ_I, ϕ_C, and ϕ_L measures, respectively.

From the figure, we observe two things. First, the leakage for some attributes such as "Name", "Location", and "Homepage" is higher than the leakage for "Relationship" and "Birthdate" (regardless of which measure is observed). This observation can be explained using Table 15, where we see that the attributes with low leakage are the ones that are revealed in a small number of social networks. This confirms our hypothesis that the overall attribute leakage is related to the number of social networks that reveal the attribute. Second, we observe that the inconsistency (i.e., the difference between the ϕ_U and ϕ_L values) for some attributes (e.g., "Name" and "Location") is much higher than the inconsistency of other attributes (e.g., "Relationship" and "Sex"). These inconsistencies are due to the large number of possible values for "Name" and "Location" (thousands) versus the small set of possible values offered to the user for "Relationship" and "Sex" (usually less than 5). This imbalance of possible values also made our standardization process for "Name" and "Location" less effective than for attributes with a smaller set of possible values.

Although this initial overview is useful, it combines users with a different number of social networks into one group. To get a deeper understanding of how the attribute leakage varies as the number of social networks an attacker finds increases, we also investigate the

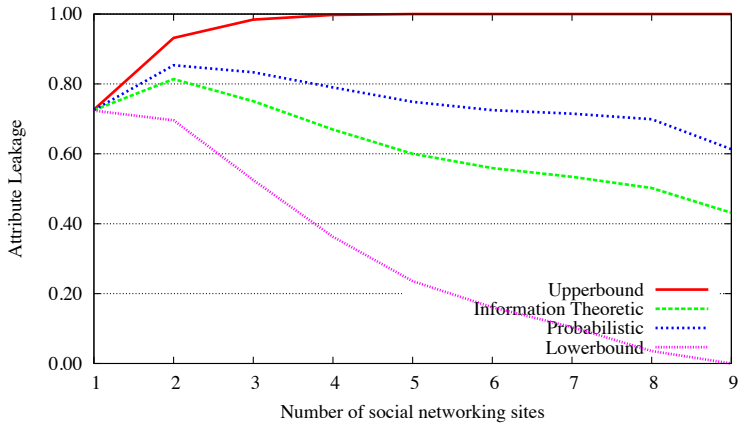


Figure 28: Attribute Leakage measures for "Name" (ϕ_i("Name", P)). Different attribute leakages (ϕ_i) are represented by different lines.

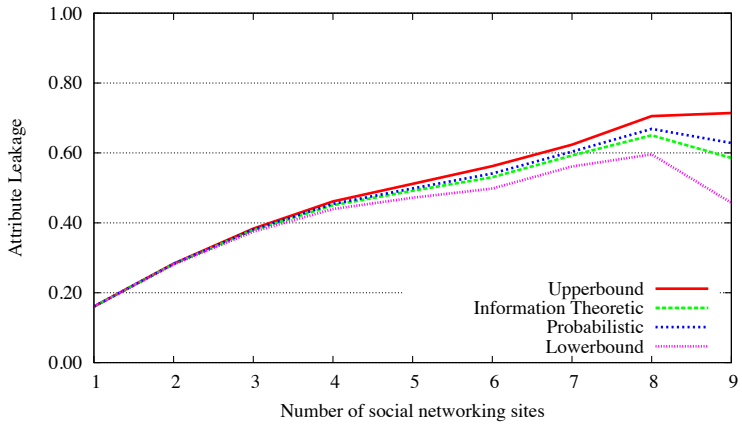


Figure 29: Attribute Leakage measures for "Hometown" (ϕ_i("Hometown", P)). Different attribute leakages (ϕ_i) are represented by different lines.

attribute leakage as it relates to the number of social networks. To include cases where an attacker may only discover a subset of social networks that belong to a user with x social networks, we include all users where $y \leq x$ when looking at the attribute leakage for y social networks. Specifically, we pick every combination of y social networking sites from the user's online identify and calculate the average of the attribute leakage metric across combinations. This results in $\binom{x}{y}$ combinations. We pick this method because it avoids introducing biases that any pre-determined ordering might impose. Henceforth, we employ this technique in all figures that plot leakage against the number of social networks belonging to a user's social footprint.

The attribute leakage, varying based on the number of social networks observed, for the

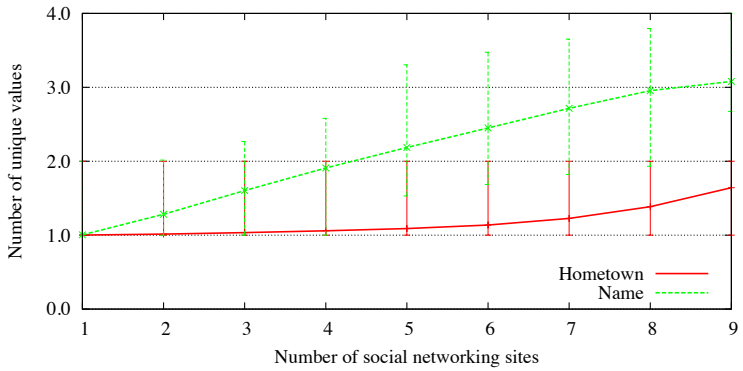


Figure 30: Number of unique values (y-axis) for "Hometown" and "Name" attributes, as the number of social networking sites in a user's online footprint increases (x-axis).

"Name" and "Hometown" attributes are shown in Figures 28 and 29, respectively. In both figures, the y-axis represents the average attribute leakage for users with the number of social networking sites shown on the x-axis. We investigated the "Name" attribute because it represents an attribute with varying attribute leakage measures in Figure 27. Conversely, we studied the "Hometown" attribute because it exhibited mostly similar average attribute leakage measures.

For the "Name" attribute, we see that the ϕ_U attribute leakage for users with one social network is 0.73, and it quickly rises to 1.0 for users with 4 social networks. This result is expected because all but one social network reveals the "Name" attribute (as shown in Table 15). As the number of sites increases, the consistency between attribute values begins to decline, and ϕ_I, ϕ_C, and ϕ_L begin to diverge. The ϕ_I and ϕ_C measures increase slightly at two social networks due to an increase in the number of user social footprint combinations that contain the "Name" attribute. The downward trend of the ϕ_I and ϕ_C measures starting at three social network sites is linked again to the decrease in consistency between the "Name" values. An investigation of a sample of users with low ϕ_C attribute leakage showed that differences in the name formats contributed to these inconsistencies (e.g., "William Smith", "Bill", "Billy", and "Bill Smith"). Finally, the ϕ_L measure can be directly related to the average number of unique attribute values for the "Name" attribute. This is also confirmed by Figure 30, which shows the average number of unique attribute values for the "Name" and "Hometown" attributes. If we compare Figures 28 and 30, we

see that as the average number of unique attribute values increases, the ϕ_L attribute leakage decreases.

For the "Hometown" attribute, we see that the ϕ_U measure for users with one social network is 0.17, and it consistently increases until its maximum of 0.71 at 9 social networks. This result is also expected because only 5 social networks reveal the "Hometown" attribute (as shown in Table 15), and as the number of social networks increases, so does the likelihood that the value will be discoverable on a network in the user's social footprint. As the number of networks increases, the consistency between attributes is very high, and the ϕ_I, ϕ_C, and ϕ_L measures remain close to the ϕ_U measure. This observation can also be confirmed using the number of unique values of the "Hometown" attribute (as shown in Figure 30). We attribute the dip in attribute leakage measures at ϕ_I, ϕ_C, and ϕ_L to inconsistencies in attribute values for some outlying users that have a large number of social networks and the reduced statistical significance of fewer users with 9 social networks.

The "Name" and "Hometown" attributes both have ϕ_I measures that are lower than the corresponding ϕ_C measures. This indicates that an attacker would take more "smart" guesses to determine the true value of those attributes. As previously mentioned, this phenomenon can occur when you have a small set of possible guesses and it takes a larger number of expected "smart" guesses to reach the true value. This can be verified by looking at the average number of unique values for either attribute in Figure 30.

The average attribute leakage hides information about the distribution of attribute leakages that make up the average. Figure 31 shows the average information theoretic attribute leakage for the "Hometown" attribute as well as the normalized density of users that make up the average, for each number of social networking sites. The x-axis displays the number of social networking sites, while the y-axis shows the amount of attribute leakage. The shaded cells indicate the density of users that contribute to the average "Hometown" information theoretic attribute leakage, normalized across the total number of users that have x social networking sites or more. The darker the shading of a cell indicates a higher density of users in that particular attribute leakage range. The graph shows us that the majority of users lie in the [0,0.05] or (0.95, 1] ranges, with a shift of users towards the (0.95,

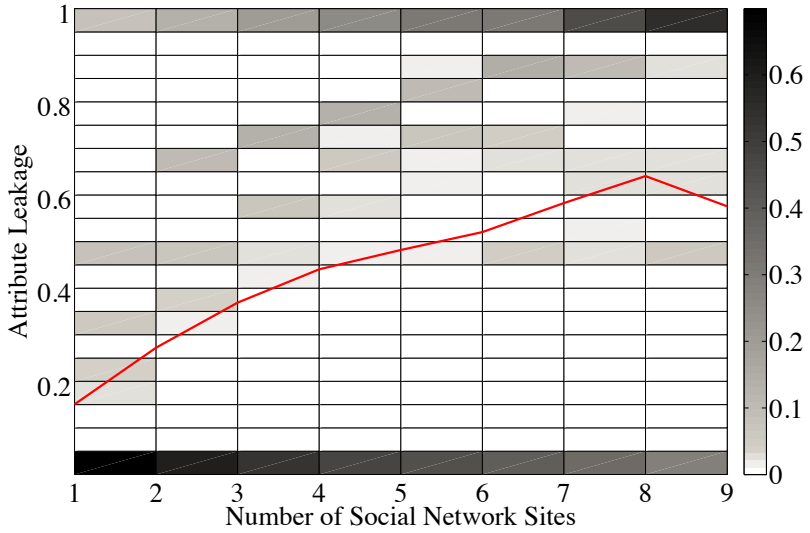


Figure 31: Normalized density of users with different hometown information theoretic attribute leakage ($\phi_I(\text{“}Hometown'', P)$). The line represents the average hometown information theoretic attribute leakage of users with x or more social network sites.

1] range as the number of social networking sites increases. As the amount of information theoretic attribute leakage increases, this figure indicates that an attacker would need to make fewer guesses to find the right value for an attribute (e.g., if a user with 3 social networks has an average ϕ_I measure of 0.37, an attacker must make an average of 2.7 “smart” guesses to get the right value for the attribute; if a user with 6 social networks has an average ϕ_I measure of 0.53, an attacker must make an average of 1.9 guesses to get the right value for the attribute[1]).

So far we have used a number of measures to look at the attribute leakage for a particular attribute. This has helped confirm our hypotheses and make interesting observations, which were originally presented using the definitions of the measures. In the next section, we investigate the aggregate attribute leakage measures to determine the leakage of a set of attributes in relation to particular attacks.

[1]The denominators of both ϕ_I and ϕ_C are intuitively the number of guesses it would take an attacker to guess the right value for an attribute. Thus, taking the reciprocal of 0.37 and 0.53 gives us the number of guesses.

7.5.2 Aggregate Attribute Leakage Measures

The aggregate attribute leakage measures offer a combined view of the information leaked by a set of attribute required for a particular attack (as defined in Section 7.1). We begin by looking at the overall aggregate attribute leakage measures, shown in Figure 32, to quantify the average exposure of the users in our dataset to the particular attacks. The x-axis represents the different attacks, while the y-axis represents the average aggregate attribute leakage. For Figures 32-33, the Upperbound, Information Theoretic, Probabilistic, and Lowerbound labels represent the Ψ_U, Ψ_I, Ψ_C, and Ψ_L measures, respectively. The Ψ_U for both attacks is above 0.5, which means that on average, users reveal more than half of the attributes required for those attacks. Although the average aggregate attribute leakage measures Ψ_I and Ψ_C still take into account consistency differences between attribute values revealed, the intuitive meaning—being the number of guesses to find the right value for a set of attributes—is lost. A social footprint whose $\phi_I(\text{“Birthdate”}, P) = 0$, $\phi_I(\text{“Location”}, P) = 0$, and $\phi_I(\text{“Gender”}, P) = 1$ has an average identification aggregate attribute leakage of 0.33. The incorrect intuition derived from this result would be that ∼3 questions would be needed to guess the value of each attribute. However, since two of the three attributes are not leaked at all, these values cannot be guessed. If viewed in relation to Ψ_U, which would be 0.33 in the above example, we would know that only one attribute on average is leaked as the Ψ_U would equal Ψ_I.

Once again, although this initial view provides a good overview of the different measures, we would like to see how the aggregate attribute leakage changes as the number of social networks discovered by an attacker increases. This would also allow us to test our hypothesis that the aggregate attribute leakage increases as the number of social networks increases. Figure 33 shows the average aggregate attribute leakage, represented on the y-axis, as the number of social networking sites an attacker discovers (shown on the x-axis) increases. From this figure, we observe that the upperbound average identification aggregate attribute leakage is 0.34 (average 1 attribute of 3 attributes) with one social network, which increases to over 0.9 (average 2.7 attributes of 3 attributes) with six or more social networks. Taking into account that the upperbound aggregate attribute leakage simply

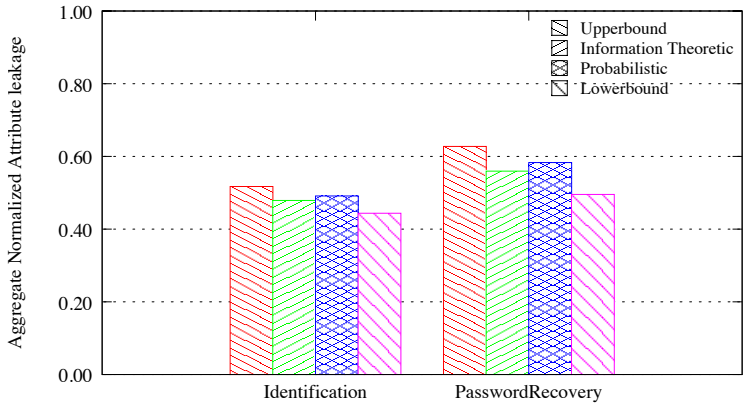


Figure 32: Aggregate Normalized Attribute Leakage ($\Psi_i(F_a, P)$) for defined attacks. Each attack is represented by a set of clustered bars on the x-axis and different aggregate normalized attribute leakages (Ψ_i) by individual bars.

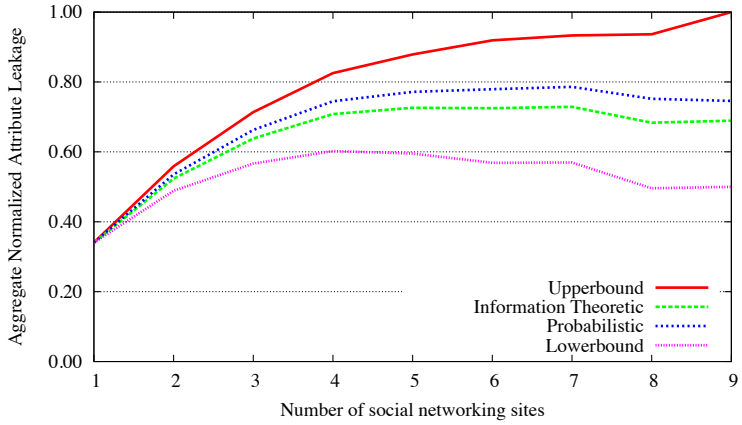


Figure 33: Aggregate Normalized Attribute Leakage ($\Psi_i(F_a, P)$) for the Identification attack. Different aggregate attribute leakages (Ψ_i) are represented by different lines.

checks for the presence of a attribute, this result implies that with one social network a person reveals 1 attribute of data required (on average) for the identification aggregate attribute leakage attack. However, with more than 6 social networks, a person reveals an average of 2.7 attributes. The lowerbound identification aggregate attribute leakage peaks at 0.6 with four social networks, and then, it declines. This occurs because location, one of the attributes needed for identification leakage, has varying representations on various social networks, and as the number of social networks increases, these inconsistencies increase (even after our standardization efforts). We find that our hypothesis is supported by the upperbound measure, which shows that the amount of information an attacker discovers increases with an increase in the number of social networks, but is contradicted by the

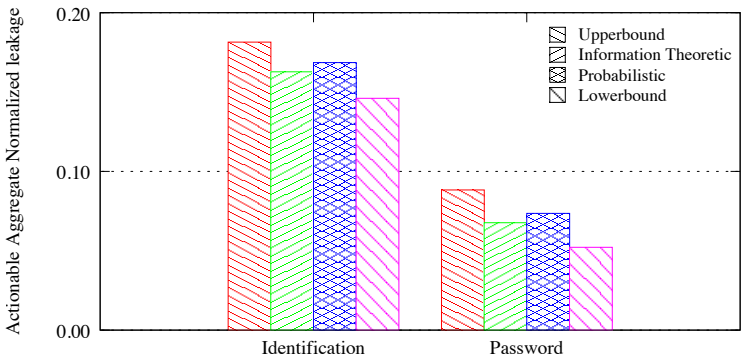


Figure 34: Actionable Aggregate Normalized Attribute Leakage ($\Theta_i(F_a, P)$) for defined attacks. Each attack is represented by a set of clustered bars on the x-axis and different aggregate normalized attribute leakages (Θ_i) by individual bars.

consistency measures (Ψ_I, Ψ_C, and Ψ_L), which show that this increase in information is offset by inconsistencies between attribute values causing the average identification aggregate attribute leakage to decrease past a certain number of social networking sites.

Although Figure 33 shows the average aggregate attribute leakage, it does not show the actionable aggregate attribute leakage (i.e., the number of users whose social footprints contained all of the attributes). Figure 34 shows the different actionable aggregate attribute leakage measures, grouped by attack with the Upperbound, Information Theoretic, Probabilistic, and Lowerbound labels represent the Θ_U, Θ_I, Θ_C, and Θ_L measures, respectively (this convention is also used for Figure 35. From this figure, we observe that the Θ_U measures for the identification and password recovery attacks are 0.18 and 0.09, respectively. This means that 18% and 9% of all users in our dataset reveal at least some information for all the attributes necessary for the identification and password recovery attacks, respectively. The identification attack has the highest actionable aggregate attribute leakage because it only requires 3 fairly common attributes to be revealed. In comparison, the password recovery attack contains 8 attributes, and some of them are fairly sensitive and only revealed on one social networking site.

Figure 35 shows the actionable identification aggregate normalized attribute leakage, represented on the y-axis, as the number of social networks a user has (shown on the x-axis) increases. We use this figure to determine how the actionable aggregate attribute leakage

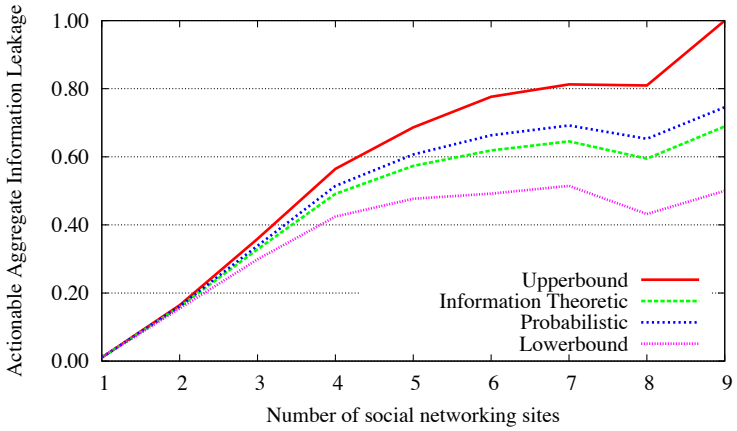


Figure 35: Actionable Aggregate Normalized Attribute Leakage ($\Theta_i(F_a, P)$) for the Identification attack. Different actionable aggregate attribute leakages (Θ_i) are represented by different lines.

changes as the number of social network sites discovered increases and if this matches the change of the average aggregate attribute leakage as the number of social network sites discovered increases. We see that even with one social networking site, a number of users (average of 0.02, which translates to 2% of users) reveal all the attributes necessary for an identification attack. The upperbound increases to about 90% (average of 0.9, which translates to 90% of users), indicating that the higher the number of social networks a user belongs to, the higher the likelihood of them being vulnerable to this attack. Another way to explain this is that as the number of social networks increases, it is more likely for a user to have a profile on a social network where this information is revealed. The Θ_I and Θ_C measures are similar to Ψ_I and Ψ_C in that they do not retain their constituent attribute leakage measure's intuitive meaning (i.e., calculating the average number of guesses required to guess the true value for a set of attributes will not equal the average of the number of guesses of the constituent attributes). Thus, even Θ_I and Θ_C should only be used in relation to Θ_U. We also see the probabilistic and information theoretic leakage decrease slightly around 8 social networking sites. This, once again, is due to location being revealed on multiple social network sites and the attribute value not being consistent.

We have used the aggregate attribute leakage measures to quantify the average and actionable aggregate attribute leakage. Doing so in a number of ways showed the value of the aggregate attribute leakage measures, while confirming our original hypotheses. We

submit that the Ψ_U and Θ_U bounds are likely of most interest when looking to quantify the likelihood of a threat. Also, taking the Ψ_C, Ψ_I, Θ_C, and Θ_I measures in relation to the Ψ_U and Θ_U bounds, respectively, is the appropriate way to look at the consistencies of the information required in these attacks.

7.6 Applying Cloaking to Counter Information Leak

One solution to the aggregate attribute leakage problem is for social networks to limit or cloak attributes released to non-friends or unauthenticated users on the social network site. Social network services often prefer to expose profile information of users to enable social activities such as linking users with common interests, but precise information is often unnecessary.

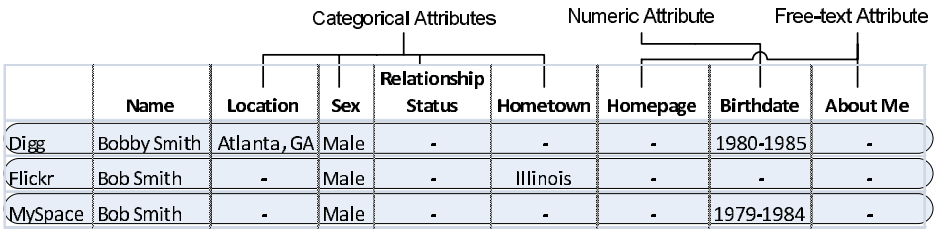
Popular approaches to anonymizing data, including k-anonymity or l-diversity, do not directly apply to social networks because in social networks, each user needs to be identifiable (most commonly using a name and picture) so that others on the social network can add or interact with such individual users. By cloaking "sensitive" attributes released to non-friends or unauthenticated users, we want to limit the attribute leakage without affecting identifiability.

7.6.1 Cloaking Attributes

Cloaking is an operation to reduce the amount of information revealed by a certain attribute. The cloaking operation requires semantic information to be associated with the attribute being cloaked to allow controlled reduction of information. We define three main types of attributes and associated cloaking operations:

Numeric attributes: To cloak numeric attributes, we release a range of values rather than a single value. To do this, one would require pre-defined ranges to be specified for the particular attribute. On cloaking the attribute for a particular user, the range would be revealed instead of the specific value of the attribute. For example, for the attribute "age", the social network would have pre-existing ranges for the age (e.g., 15-19, 20-24) that could be revealed instead of the actual value.

Categorical attributes: To cloak categorical attributes, semantic information about a



	Name	Location	Sex	Relationship Status	Hometown	Homepage	Birthdate	About Me
Digg	Bobby Smith	Atlanta, GA	Male	-	-	-	1980-1985	-
Flickr	Bob Smith	-	Male	-	Illinois	-	-	-
MySpace	Bob Smith	-	Male	-	-	-	1979-1984	-

Figure 36: User Bob Smith's Online Social Footprint (from Figure 20), after being cloaked.

hierarchy of possible values would be required. Instead of releasing the user's specific attribute value, the attribute at the higher level (given that each parent level has more than one child level) would be released. An example of this would be the "location" attribute, which could range from a street name, to a zip code, to a municipal name, to the name of a city, and finally, to the state. Some attributes may not have semantic information associated with them or may not possess a parent; in such cases, the cloak operation would simply suppress the attribute.

Free-text attributes: Free-text attributes are harder to cloak as the cloak operation would have to perform "summarization" to reduce the amount of free-text when the attribute is being cloaked. We do not implement the cloak operation in this manner and simply suppress such attributes.

7.6.2 Attribute Leakage with Cloaking

Attribute leakage is used as a core part of the calculation of aggregate attribute leakage, and as a result, we show the effect of cloaking on the different attribute leakage ($\phi_i(f_a, P)$) measures.

$\phi_U(f_a, P)$, will remain unchanged if any number of social networks cloak their attributes because this is the upper-bound, which simply checks if the value is released or not. $\phi_C(f_a, P)$ relies on the list of attribute values revealed by a user's social footprint. Assuming the user's social footprint contains only one social network and cloaking the attribute introduces r different values (e.g., a numeric attribute is cloaked to a range of r values or a categorical attribute is cloaked to a parent that has at least r sub-values), then instead of 1 value for the attribute, there would be r possible values for the attribute (each

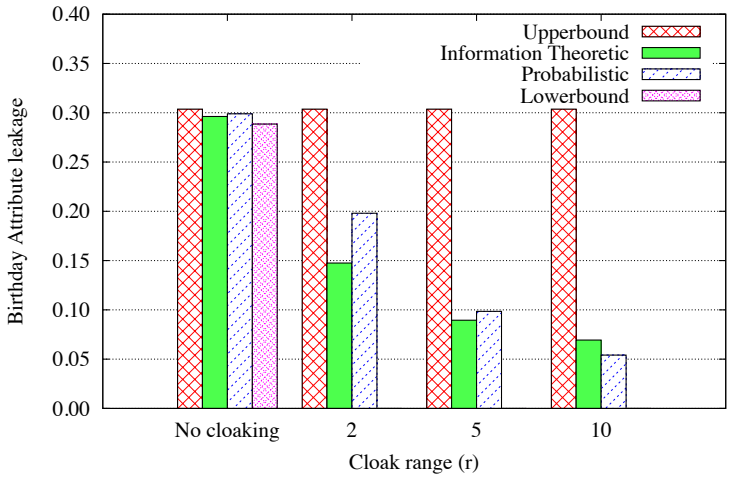


Figure 37: Attribute Leakage of Birthdate, after being cloaked. Different cloak ranges are shown on the x-axis with the resulting attribute leakage on the y-axis.

with an equal probability). The denominator (or the number of guesses required) would be $\frac{r+1}{2}$ (we omit the straight-forward derivation due to space-constraints), resulting in an attribute leakage of $\frac{2}{r+1}$. For example, if r is 5, the expected number of guesses required would be 3, and the resulting attribute leakage would be $\frac{1}{3}$. $\phi_I(f_a, P)$ behaves similar to $\phi_C(f_a, P)$. Using the previous assumptions the resulting denominator (or number of "smart" guesses required) is $1 + log\frac{1}{r}$. Once again, with any cloak range r greater than 1, this resulting value will be greater than 1. For a cloak range of 5, the expected number of guesses required would be 3.3, and the resulting attribute leakage would be 0.30. There are a number of other complex cases for $\phi_I(f_a, P)$ and $\phi_C(f_a, P)$, but in each case, the attribute leakage monotonically decreases. $\phi_L(f_a, P)$ will always be zero even if one social network releases the attribute in its cloaked form.

Figure 36 shows the result of cloaking the user Bob Smith's information from Figure 20. The free-text attributes are completely suppressed, and the categorical attributes Location and Hometown are cloaked to the city and state level, respectively. The birthdate attribute is converted to birthyear and cloaked as a numeric attribute.

7.6.3 Experimental Evaluation of Cloaking

For the experimental evaluation of cloaking attributes, we look at the birthdate attribute because it is one of the attributes that we have defined the operation over.

The cloaking operation for birthdate strips off the birthday and birthmonth, leaving only the birthyear. Additionally, we cloak the year by providing a range of r values instead of a single value. We use the same cloak range for each social network site, but the ranges differ based on the birthyear of the youngest profile present on the site. As an example, the range of years (when the cloak range size is 5) for an individual 21 years of age is 20-24, 19-23, 21-25, and 21-25 for Digg, LiveJournal, MySpace, and YouTube, respectively.

Figure 37 shows the average attribute leakage (y-axis) of cloaking the birthdate attribute with different cloak ranges, represented by a cluster of bars on the x-axis, over the entire dataset. "No cloaking" represents the original attribute leakage without any cloaking. As expected, the Upperbound Attribute leakage is not affected by cloaking and the Lowerbound Attribute leakage is zero if cloaking is applied (regardless of cloak range). Both the Information Theoretic Attribute leakage and the Probabilistic Attribute leakage reduce as the size of the range increases, due to more expected guesses needing to be made to arrive at the right value. The Information Theoretic Attribute leakage becomes higher than the Probabilistic Attribute leakage after the range is increased to 8 (not shown).

7.7 Related Work

Krishnamurthy et al. [60] present statistics on privacy settings of two popular social networks. They show that 79% of MySpace allowed their profile, friends, comments, and user content to be viewable. Sophos [88] similarly looks at the default privacy settings of Facebook and finds them quite lax. From a sampling of 200 users of 1.2 million users in the London regional network they find 54% show their full birthdate and 1% divulge their phone number. Gross et al. [41], in addition to showing that Facebook users at CMU do not use the privacy limiting preferences, contains an excellent discussion about the real world implications and ways in which social network information can be misused. Based on CMU users on Facebook, they provide statistics about participation, information validity, as well as types and amount of information revelation. Chew et al. [27], compliment this discussion by providing examples on how "Activity Streams", "Unwelcome Linkage", and "Merging Social Graphs", can compromise a user's privacy. Mislove et al. [74] study the graphs of

multiple social networks, providing interesting measurements and observations of their the stucture.

Anonymization of public datasets containing sensitive attributes has been studied extensively with techniques such as k-anonymity [96], ℓ-diversity [67], and t-closeness [66]. Each of these techniques aims to generalize or suppress quasi-identifiers in-order to prevent linkage to secondary datasets or prior knowledge. This is done to protect the "sensitive" attributes released along with the quasi-identifiers. In each case, the benefit of using such anonymization techniques is the controlled reduction of utility in relation to privacy.Our aims differ from these as we propose cloaking the sensitive attributes released publicly (to non-friends and unauthenticated users) in an effort to reduce a user's unintended personal information leakage.

7.8 Conclusion

In this chapter, we have defined quantitative measures that provide varying bounds on unintended personal information leakage. Using approximately 8,200 users' social footprints from an online identity management site, we study the amount of aggregate attribute leakage in our dataset making observations that would not have been possible without our measures. We also use the measures to verify a number of hypotheses, in regards to the amount of personal information being discoverable as the number of social network sites a user belongs to increases. Concretely, looking at the identification attack, we find that only 2% of users are vulnerable when looking at one social site, but this increases to 18% when looking at a person's social footprint. The password recovery attack follows a similar trend. We show cloaking can effectively be used to reduce the amount of information an attacker can obtain from a person's social footprint. Using the "Birthdate" attribute, we show a reduction of 66% for the Information Theoretic and Probabilistic attribute leakage measures when cloaking is applied (with a cloaking range of 5).

CHAPTER VIII

REVERSE SOCIAL-ENGINEERING

Social networking sites such as Facebook, LinkedIn, and Twitter are arguably the fastest growing web-based online services today. Facebook, for example, has been reporting growth rates as high as 3% per week, with more than 400 million registered users as of March 2010 [15]. Many users appreciate social networks because they make it easier to meet new people, find old friends, and share multimedia artifacts such as videos and photographs.

One of the key features of social networks is the support they provide for finding new friends. For example, a typical technique consists of automatically identifying common friends in cliques and then promoting new friendships with messages such as "*You have 4 mutual friends with John Doe. Would you like to add John Doe as a new friend?*". Also, information on the activities of users are often collected, analyzed, and correlated to determine the probability that two users may know each other. If a potential acquaintance is detected, a new friendship recommendation might be displayed by the social network site when the user logs in.

Clearly, social networks are critical applications with respect to the security and privacy of their users. In fact, the large amount of information published, and often publicly shared, on the user profiles is increasingly attracting the attention of attackers. Attacks on social networks are usually variants of traditional security threats (such as malware, worms, spam, and phishing). However, these attacks are carried out in a different context by leveraging the social networks as a new medium to reach the victims. Moreover, adversaries can take advantage of the trust relationships between "friends" in social networks to craft more convincing attacks by exploiting personal information gleaned from victims' pages.

Past research has shown that users of online social networks tend to exhibit a higher degree of trust in friend requests and messages sent by other users (e.g., [13,23]). In addition, some forms of attacks on social networks, such as the problem of unsolicited messages, have

already been studied in detail by the research community (e.g., [47, 93]). However, to date, *reverse social engineering* attacks in social networks have not received any attention. Hence, no previous work exists on the topic.

In a reverse social engineering attack, the attacker does not initiate contact with the victim. Rather, the victim is tricked into contacting the attacker herself. As a result, a high degree of trust is established between the victim and the attacker as the victim is the entity that first wanted to establish a relationship. Once a reverse social engineering attack is successful (i.e., the attacker has established a friend relationship with the victim), she can then launch a wide range of attacks such as persuading victims to click on malicious links, blackmailing, identity theft, and phishing.

This chapter presents the first user study on how attackers can abuse some of the features provided by online social networks with the aim of launching automated reverse social engineering attacks. We present three novel attacks, namely, recommendation-based, visitor tracking-based, and demographics-based reverse social engineering. Furthermore, using the popular social networks Facebook, Badoo, and Friendster, we discuss and measure the effectiveness of these attacks, and we show which social networking features make such attacks feasible in practice.

In the recommendation attack, the aim is to exploit the friend recommendations made by the social network to promote the fake profile of a fictitious user to the victim. The hope, from the attacker's point of view, is that the victim will be intrigued by the recommendation, and will attempt to contact the bogus profile that is under the attacker's control. In the visitor tracking attack, the aim is to trigger the target's curiosity by simply browsing her profile page. The notification that the page has been visited may be enough to attract the target to visit the attacker profile. Finally, in the demographic-based attack scenario, the attacker attempts to reach his victims by forging fake demographic or personal information with the aim of attracting the attention of users with similar preferences (e.g., similar musical tastes, similar interests, etc.).

Our findings suggest that, contrary to the common folk wisdom, only having an account with an attractive photograph may not be enough to recruit a high number of unsuspecting

victims. Rather, the attacker needs to provide victims with a pretext and an incentive for establishing contact.

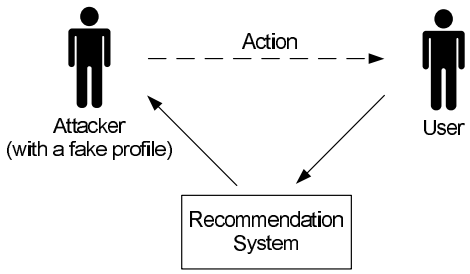
In this chapter, we make the following contributions:

- We present the first user study on reverse social engineering in social networks and present three novel attacks. In particular, we discuss and measure how attackers can abuse some of the friend-finding features that online social networks provide with the aim of launching automated reverse social engineering attacks against victims.
- We measure how different user profile attributes and friend recommendation features affect the success of reverse social engineering attempts.
- We study the interactions of users with accounts that have been set up to perform reverse social engineering, and provide insights into why users fall victim to such attacks.
- We propose mitigation techniques to secure social networks against reverse social engineering attempts.

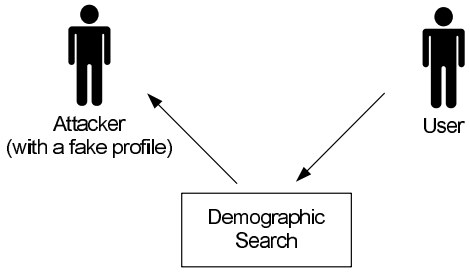
8.1 Reverse Social Engineering in Social Networks

Online social engineering attacks are easy to propagate, difficult to trace back to the attacker, and usually involves a low cost per targeted user. They are well-known threats in which the attacker aims at influencing the victims, and making them perform actions on her behalf. The attacker is typically interested in tricking the victims into revealing sensitive or important information. Examples of these attacks include traditional e-mail hoaxes and phishing, or their more advanced targeted forms, such as spear phishing.

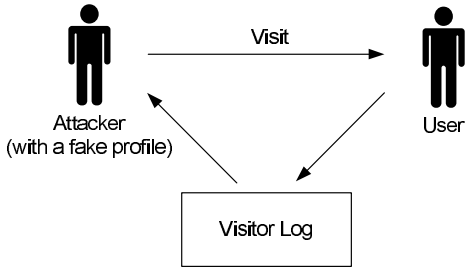
Most online social engineering attacks rely on some form of "pretexting" [75]. That is, the attacker establishes contact with the target, and sends some initial request to bootstrap the attack. This approach, although effective because it can reach a large number of potential victims, has the downside that Internet users are becoming more and more suspicious about unsolicited contact requests. However, previous work has shown that it is possible



(a) Recommendation Systems



(b) Demographic Search



(c) Visitor Tracking

Figure 38: Different types of Reverse Social Engineering attacks.

to raise levels of trust by impersonating an existing friend of the target (e.g., [23, 50]) or by injecting the attack into existing chat conversations [62].

Reverse Social Engineering (RSE) is a form of social engineering attack that has not yet been reported widely in an online context. RSE is a well-known technique in the hacker community (e.g., [75]) for targeted phone attacks. The attack, in a first step, relies on some form of "baiting" to stimulate the victim's curiosity. In a second step, once the victim's interest is raised, the attacker waits for the victim to make the initial approach and initiate contact. An RSE attack usually requires the attacker to create a persona that would seem attractive to the victim and that would encourage the victim to establish contact. For example, directly calling users and asking them for their passwords on the phone might

raise suspicion in some users. In the reverse social engineering version of the same attack, a phone number can be e-mailed to the targets a couple of days in advance by spoofing an e-mail from the system administrator. The e-mail may instruct the users to call this number in case of problems. In this example, any victim who calls the phone number would probably be less suspicious and more willing to share information as she has initiated the first contact.

RSE attacks are especially attractive for online social networks. First, from an attacker's point of view, there is a good potential to reach millions of registered users in this new social setting. Second, RSE has the advantage that it can bypass current behavioral and filter-based detection techniques that aim to prevent wide-spread unsolicited contact. Third, if the victim contacts the attacker, less suspicion is raised, and there is a higher probability that a social engineering attack (e.g., phishing, a financial scam, information theft, etc.) will be successful.

In general, Reverse Social Engineering attacks can be classified based on two main characteristics:

- *Targeted/Un-targeted*: In a targeted attack, the attacker focuses on a particular user. In contrast, in an un-targeted attack, the attacker is solely interested in reaching as many users as possible. Note that in order to perform a targeted attack, the attacker has to know (or acquire) some previous information about the target (e.g., such as her username or e-mail address).

- *Direct/Mediated*: In a direct attack, the baiting action of the attacker is visible to the targeted users. For example, an attacker can post a message on a public forum, or publish some interesting picture on a website. Mediated attacks, in contrast, follow a two-step approach in which the baiting is collected by an intermediate agent that is then responsible for propagating it (often in a different form) to the targeted users.

In the following, we present three different combinations of RSE attacks within the context of online social networks.

8.1.1 Recommendation-Based RSE

Characteristics: [Targeted, Mediated]

Recommendation systems in social networks propose relationships between users based on background, or "secondary knowledge" on users. This knowledge derives from the interactions between registered users, the friend relationships between them, and other artifacts based on their interaction with the social network. For example, the social networking site might record the fact that a user has visited a certain profile, a page, a picture, and also log the search terms she has entered. Popular social networks (e.g., Facebook) often use this information to make recommendations to users (e.g., "*Visit page X*", "*You might know person Y, click here to become her friends*", etc.).

From an attacker's point of view, a recommendation system is an interesting target. If the attacker is able to influence the recommendation system and make the social network issue targeted recommendations, she may be able to trick victims into contacting her. Figure 38(a) demonstrates the recommendation system-based RSE attack scenario.

8.1.2 Demographic-Based RSE

Characteristics: [Un-targeted, Mediated]

Demographic-based systems in social networks allow establishing friendships based on the information in a person's profile. Some social networks, especially dating sites (e.g., Badoo), use this technique as the norm for connecting users in the same geographical location, in the same age group, or those who have expressed similar preferences.

Figure 38(b) demonstrates an RSE attack that uses demographic information. In the attack, the attacker simply creates a profile (or a number of profiles) that would have a high probability of appealing to certain users, and then waits for victims to initiate contact.

8.1.3 Visitor Tracking-Based RSE

Characteristics: [Targeted, Direct]

Visitor tracking is a feature provided by some social networks (e.g., Xing, Friendster) to allow users to track who has visited their online profiles.

Table 16: RSE attacks on three popular social networks. ✓ indicates that the attack is possible; ✠ indicates that we demonstrate and measure the effectiveness of this attack on the particular social network.

Type of Attack	Facebook	Badoo	Friendster
Recommendation-Based	✓✠	-	-
Demographic- Based	✓	✓✠	✓
Visitor Tracking-Based	-	✓	✓✠

The attack in this case involves exploiting the user's curiosity by visiting their profile page. The notification that the page has been visited might raise interest, baiting the user to view the attacker's profile and perhaps take some action. Figure 38(c) outlines this attack method.

8.2 RSE Attacks in the Real-World

In this section, we present three types of real-world RSE attacks that are possible on three different social network platforms: Facebook, Badoo, and Friendster. In particular, we describe a recommendation-based RSE attack on Facebook, a demographic-based RSE attack on Badoo, and a visitor tracking-based RSE attack on Friendster.

Table 16 shows the social networks that were used in our experiments, and also describes which kind of RSE attacks are possible against them. Note that not all the combinations are possible in practice. For example, Facebook does not provide any information about the users that visit a certain profile, thus making a visitor tracking attack infeasible. In the rest of this section, we describe the different steps that are required to automate the attacks, and the setup of the experiments we performed.

8.2.1 Ethical and Legal Considerations

Real-world experiments involving social networks may be considered an ethically sensitive area. Clearly, one question that arises is if it is ethically acceptable and justifiable to conduct experiments that involve real users. Similar to the experiments conducted by Jakobsson et al. [51, 52] and our previous work [23], we believe that realistic experiments are the only way to reliably estimate success rates of attacks in the real-world.

Furthermore, during all the experiments we describe in the chapter, we took into account

the privacy of the users, and the sensitivity of the data that was collected. When the data was analyzed, identifiers (e.g., names) were anonymized, and no manual inspection of the collected data was performed.

Note that all the experiments described in the chapter were performed in Europe. Hence, we consulted with the legal department of our institution (comparable to the Institute Review Board (IRB) in the US) and our handling and privacy precautions were deemed appropriate and consistent with the European legal position.

8.2.2 Influencing Friend Recommendations

A good example of a real recommendation system is Facebook's friend suggestions. During our tests with Facebook, we observed that Facebook promotes the connection of users by suggesting them friends that they probably know. The system computes these suggestions based on common information, such as mutual friends, schools, companies, and interests. This feature is well-known to many social network users. In fact, whenever a user is logged in, she is regularly notified of persons that she may know.

Previous work [18] has shown that Facebook also uses the e-mail addresses a user has queried to identify a possible friendship connection between two users. The premise is that if users know each other's e-mail addresses, they must be connected in some way. Therefore, if an attacker gains access to the e-mail address of a victim (e.g., a spammer who has a list of e-mails at her disposal), by searching for that address, she can have a fake attacker profile be recommended to the victims. In our experiments, we observed that this technique results in the attacker profile being the most highly recommended profile.

For the first experiment, we used the data collected for over a year in a previous study we performed on Facebook [18]. In the study, we registered a single account that we used to perform a large number of e-mail search queries, using an email list obtained from a dropzone on a machine compromised by attackers. Without our knowledge, our profile was later recommended to all the queried users as a potential friend. As a result, our test account received thousands of messages and friend requests.

Table 17: Characteristics of the dummy profiles used in the experiments

Attribute	Prof. 1	Prof. 2	Prof. 3	Prof. 4	Prof. 5
Age	23	23	23	35	23
Sex	Male	Female	Female	Female	Female
*Location**	N.Y.	N.Y.	Paris	N.Y.	N.Y.
Real Picture	Yes	Yes	Yes	Yes	No

8.2.3 Measuring RSE Effects by Creating Attack Profiles

In the second set of experiments, we created five different attack profiles in three social networks. The profiles were designed with different characteristics to enable us to observe and measure the effects that each characteristic had on the effectiveness of the RSE attacks. That is, we were interested in determining which features would attract the higher number of potential victims using the recommendation-based, demographic-based, and visitor tracking attacks.

The five attack profiles are shown in Table 17. For the profile pictures, we used popular photographs from Wikipedia, licensed under the Creative Commons license. All photos represented an attractive male or female, with the exception of Profile 5 for which we used a synthetic cartoon picture.

Table 18 shows the number of users we targeted in the social networks we tested. For example, in the Facebook experiment, we targeted a total of 250,000 profiles, equally divided between the 5 attack profiles. In the demographic-based attack on Badoo, no action was required on behalf of the attacker. Hence, the number of targeted users is not given (i.e., all registered Badoo users could have found and contacted the attacker profile).

Table 18: Overview of OSNs as well as number of users targeted.

Social Network	# of Targets	Total users	Alexia Rank
Badoo	-	73 million	143
Facebook	250,000	500 million	2
Friendster	42,000	8.2 million	643

8.2.4 Automating the Measurement Process

During our study we developed a number of scripts to automate the three attacks and the measurement process on the different social networks.

8.2.4.1 Recommendation-Based RSE on Facebook

As shown in Figure 38(a), the recommendation-based RSE attack against Facebook consisted of two parts: First, the target user's profile was probed using an e-mail lookup, and second, the attack accounts were automatically monitored for victims who contacted these accounts based on the friendship recommendation made by Facebook.

For the first part, we used the "contact import" functionality provided by Facebook and the API provided by Google Mail's address book to automatically search for users by their e-mail addresses. We broke the total set of users we wished to query into smaller sets, and sent multiple requests to Facebook, as they have limited the number of e-mail addresses that can be queried using a single request (because of recommendations made in previous work [18]).

In the second part of the experiments, we wrote an API that allowed us to interact with Facebook to accept friend requests, fetch user profiles, as well as fetch any private message that may have been sent to the attack profiles.

Note that CAPTCHAs in Facebook were only encountered if we were not careful about rate limiting.

8.2.4.2 Demographic-Based RSE on Badoo

We used Badoo to test the demographic-based RSE attack. Hence, we only had to create the attack profiles and automatically monitor incoming connections. Just like in the recommendation-based RSE attack, we automatically retrieved and collected any message sent to the attacker profiles. Furthermore, as Badoo allows to see which users have visited a profile, we also logged this information.

8.2.4.3 Visitor Tracking-Based RSE on Friendster

We used Friendster to perform the RSE attack based on visitor tracking. As shown in Figure 38(c), this attack consists of two parts: First, we visit the target user's profile and as a consequence, the system shows to the victim that someone has visited her profile. If the attacker profile is interesting, the victim may choose to contact the attacker. Hence, in a

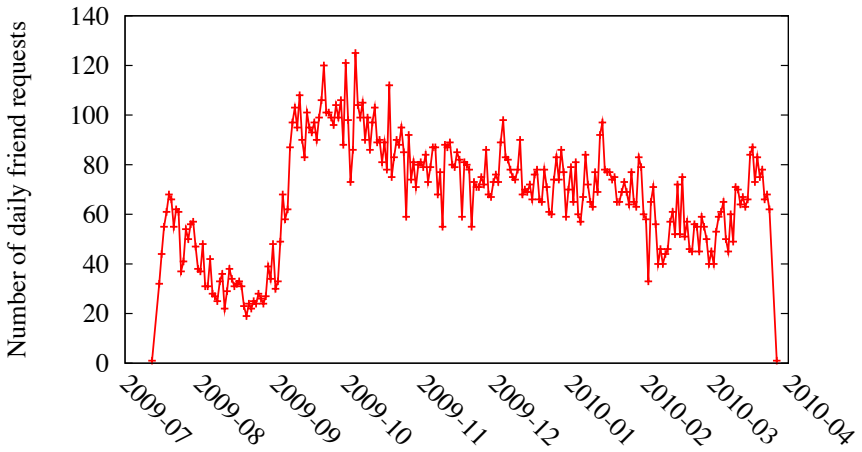


Figure 39: Daily number of new friend requests in the initial Facebook experiment

second step, the visits and the incoming messages to the attack profiles were automatically monitored to determine which of the victims came back and initiated contact.

8.3 Experimental Results

8.3.1 Recommendation-based RSE Attack

8.3.1.1 Initial Experiment

During the study [18] we conducted, we observed that the test account we were using to query e-mail addresses were receiving a large number of friend requests. The profile used in this attack was similar to Profile 2 described in Table 17.

Figure 39 shows the number of daily friend requests received by the account used in this initial experiment. The graph shows that during the first two months, the account received an average of 45 requests per day, followed by an increase to an average of 75 requests per day for the next 6 months.

The rapid increase in the number of request is the consequence of the cascading effect that commenced when we started accepting the incoming invitations. The fact that the account had a large number of friends built up the "reputation" of our profile. In addition, we started being advertised by Facebook to new people with whom we shared common friends.

Of the over 500,000 e-mails queried by our decoy profile, we were contacted by over 17,000 users (i.e., 3.3% friend connect rate within 9 months and 0.37% friend connect rate

per month). Note that our test account reached both the maximum number of active friend connections and the total number of pending friend requests allowed by Facebook.

8.3.1.2 Controlled, In-Depth Experiments

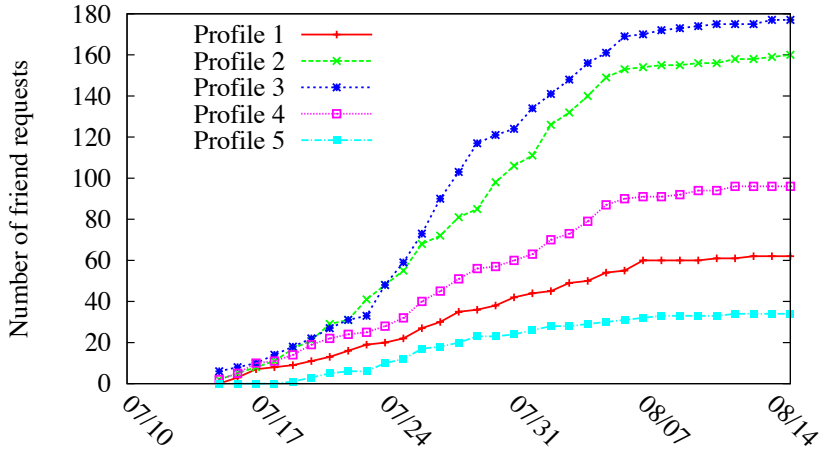
After the success of the initial experiment, we started a number of controlled, in-depth experiments to measure and determine which profile characteristics and social network features affect the success rates of RSE attacks.

To reach our goal, we created five attack profiles on Facebook. For each profile, we randomly selected 50,000 target users and looked up their e-mail addresses (hence, influencing the recommendations made by Facebook). We then measured the number of friend-requests, private messages, and other interaction sent to each attack profile. Figure 40 depicts the result of this experiment. The y-axis represents the cumulative number of friend requests or messages for the period represented by the date on the x-axis.

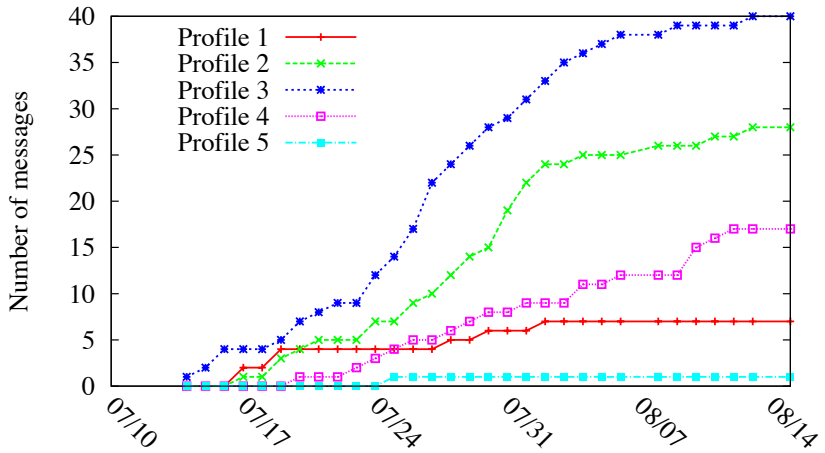
Profiles 2 and 3 were the most successful in terms of the number of friend requests and messages that were received. Both profiles correspond to attractive females who are interested in friendship. Note that there was no correlation with the location of the attack profile (i.e., the location did not influence friend requests). Hence, an initial analysis seems to confirm the general intuition that an attractive female photograph will attract potential victims. In contrast to the other profiles, Profile 5 was the least effective. In this profile, a cartoon character was used as a photograph rather than a real picture. In comparison, Profile 1 performed only slightly better than Profile 5. This profile contained the photograph of an attractive male.

Over the entire month, the most effective profile had a friend connection rate of 0.35% (i.e., in line with the initial experimental profile). The least effective profile instead, had a friend connection rate of only 0.05%.

Although friend connection requests and private messages were the most common form of interaction with a decoy profile, we also received a large number of friend suggestions. Friend suggestions are suggestions made by the victim to other users. Such suggestions are important as they imply that a high level of trust has been achieved between the attacker



(a) Friend connect requests sent to each profile



(b) Messages sent to each profile

Figure 40: Cumulative counts of interactions resulting from reverse social engineering on Facebook.

and the victim. Also, note that over 94% of the messages to the attack profiles were sent after the friend connection requests.

By analyzing the demography of the users who contacted our attack profiles, we can identify potential characteristics that make a decoy profile appealing. In particular, we focused on three fields: relationship status, interested in, and age (Figure 41). The y-axis of the figure shows the percentage of friend connection requests that originated from a profile with the respective demographic value (empty values excluded) to the attack profile listed on the x-axis. Young, single users who have expressed interest in "Women" seem to be the easiest victims to attract. In comparison, Profile 1 (the only male profile) received

a larger number of friend requests from users who had expressed interest in "Men".

Interestingly, the profile with a cartoon picture was the one to attract the largest number of requests coming from older users (i.e., those who were older than 40). Hence, the experiments show that by carefully tweaking the profile information, it is possible to obtain an higher success rate against a particular group of users.

Finally, we analyzed the messages that were sent to the different attack profiles. To protect the privacy of individuals in the study, we first processed the messages and removed user identifiers. After anonymization, we only ran word-based statistical analyses on the message contents. That is, as a pre-processing step, we used Porter's stemming algorithm on the extracted tokens [80], followed by a count of n-grams (where a single gram is a stemmed token).

Around 10% of the messages mentioned the Facebook recommendation, including 3-grams such as "suggest you as" or "suggest I add". The analysis shows that some users used the recommendation made by the social network as a pretext to contact the attack profile.

8.3.2 Demographic-based Experiment

For our demographic-based RSE attacks, we targeted Badoo, a dating oriented socializing system that allows users to meet new friends in a specific area. A registered user can list the people who have visited her profile and exchange messages with other users. Figure 42 shows the cumulative number of visitors and messages received for each attack profile we created in the network.

Profiles 2 and 3 were again the most popular, and attracted the most visitors (over 2500 each). These profiles also received the largest number of messages (i.e., more than 2500 each). Because Profile 5 was not using a photograph of a person, it was removed by Badoo from the demographic search after it was visited by 451 users and it received 383 messages. Once again, Profile 1, the attack profile of a male user, received the fewest visits and friend requests.

Another measure of how successful an attack profile was is the percentage of users who

decided to send a message after visiting a profile. These figures are over 50% for the two attractive female profiles (Profile 2 and 3), and 44% on average for all attack profiles.

We took a closer look at the demography of the users who contacted us. In the case of Badoo, sending a message is the most concrete form of interest, and one that can easily be exploited (e.g., [23]). Figure 43 shows a demographic breakdown by relationship status, what users were interested in, and age. Similar to Figure 41, the y-axis shows the percentage of users who sent messages that originated from a profile with the respective demographic value.

Note that Badoo is a site that is geared towards dating. Most of the users who initiate contact express that they are either single, or in an "open relationship". In general, the attack profiles only attracted users of the opposite gender. The age demographic shows that most of the victims belong to the same age group that the attack profile belongs to. In comparison, there was no correlation of age for contact requests on Facebook.

Another important difference with respect to Facebook was that the location was significant in Badoo. In fact, almost all the messages were sent by people living in the same country as the attack profile.

Finally, the 3-grams analysis for the messages received on Badoo showed that the most popular term was "how are you" occurring over 700 times. Other popular lines included "get to know" and "would you like", "you like" ... "chat" or "meet".

8.3.3 Visitor Tracking Experiment

In the visitor tracking RSE attack, we used each of the five attack profiles to visit 8,400 different user profiles in Friendster. As we have already previously described, on Friendster a user can check which other users have visited her profile.

In our experiment, we tracked which victims visited our attack profiles, and then counted the number of users who sent us a friend request. The results of this experiment are shown in Figure 44 (the sub-figure 44(a) and 44(b) represent the number of visitors and number of friend requests sent to the attack profiles).

The number of users who were curious about our visit, and visited us back was consistent

with the results of the experiments we conducted on other social networks (i.e., between 0.25 and 1.2% per month). However, only a few users later sent a friend request or a message.

The demographic breakdown for Friendster is presented in Figure 8.3.3. The statistical distributions are similar to the ones obtained in the Facebook experiment, showing the difference in terms of characteristics between friend-oriented and dating-oriented social networks.

8.4 Discussion and Lessons Learned

In this section, based on the results of the empirical experiments, we distill some insights about the way RSE attacks work in social networks. We can summarize our findings in two main points: The importance of having the right profile, and the importance of providing a pretext to the victims.

The first straightforward factor we were able to measure is the impact of the profile characteristics on the overall effectiveness of an attack. The experiments confirm the folk wisdom that using an attractive female photograph is a good choice to attract victims. The success rate of the most successful female profile, in terms of both friend requests and number of received messages, is between 2 and 40 times higher than the worse performing profiles (i.e., the male profile and the profile without a photograph).

Note that if the objective of the attack is not simply to reach the highest number of users, but to target a specific person or group the success rate of the attack can be improved by carefully tuning the profile characteristics. For example, our experiments show that age and location information are decisive in dating sites, while this information is not as critical in more general, friend-oriented, social networks. Also, the results suggest that gender information is always very important. Hence, a successful reverse social engineering attack should use the opposite sex of the victims in the decoy profile.

The experiments show that the impact of the profile picture is quite uniform in different social networks. For example, we observe that young users are generally more intrigued by attractive photographs, while decoy profiles (e.g., Profile 5) that do not contain the photograph of a real person tend to attract more senior users.

Obviously, even though having a catchy, interesting profile is important, our research shows that there is a second, even more important factor that contributes to the success of the attack: the pretext. Our experiments indicate that users need an incentive and a good reason to engage in interaction with a person that they do not know. In other words, users need a good excuse to "break the ice" and motivate the first approach. The differences between the success rates of the attacks on Facebook and Friendster suggest that an incentive or a pretext is critical for reverse social engineering attacks to work in practice.

The analysis of the messages received on Facebook support the hypothesis that a recommendation system gives a reason to users to initiate contact. That is, a number of users referenced the Facebook recommendation as a motivation for their friend request. In contrast, on Friendster, even though the percentage of users that browsed our decoy profiles was consistent with the other social network experiments, very few people moved to the next step and sent a contact message. The reason is, in our opinion, that the visitor tracking attack failed to provide a good pretext to the victims.

Note that the demographic experiment on Badoo was also very effective. The reason for this success is that Badoo greatly relies on the demographic search functionality to allow users to find possible contacts. In the case of a dating site, the pretext for establishing contact was the fact itself of living in a close location, or being in the same age group of the victim.

Our experiments demonstrate that reverse social engineering attacks on social networks are feasible if they are properly designed and executed. However, contrary to the common folk wisdom, only having an account with an attractive photograph may not be enough to recruit a high number of unsuspecting victims. Rather, the attacker needs to combine an attractive profile with a pretext and incentive for the victim to establish contact. Recommendation systems such as Facebook's friend suggestions are effective tools for creating such an incentive. Also, we see that profile attributes such as location and age may be the required incentives on dating networks such as Badoo.

8.5 RSE Countermeasures in OSN

Clearly, features that allow social network users to easily make new acquaintances are useful in practice. However, our chapter demonstrates that such systems may also be abused to trick users on behalf of attackers. In this section, we list three countermeasures that would increase the difficulty of launching RSE attacks in online social networks.

First, while friend recommendation features are useful, our experiments show that they may pose a risk to users if the attackers are able to somehow influence the recommendation system. Hence, it is important for social network providers to show a potential connection between two users only if there is a strong connection between them. For example, in the case of Facebook, as our experiments show, a simple e-mail lookup does not necessarily indicate that the users know each other. Thus, one could check other information, such as the fact that the users already have some friends in common.

Second, we believe that it is important to closely monitor friendships that have been established in social networks. Benign user accounts will typically send and receive friend requests in both directions. That is, a user may be contacted by people she knows, but she will also actively search and add friends on the network. However, in contrast, a honeypot RSE account (as we describe in this chapter) only receives friend requests from other users. Thus, it may be possible to identify such accounts automatically.

Third, we believe that CAPTCHA usage also needs to be extended to incoming friend requests. Today, because of the active threats of spamming and social engineering, social network providers may display CAPTCHAs when friend requests are sent to other users. However, no such precautions are taken for messages and friend requests that are received. By requiring users to solve a CAPTCHA challenge before being able to accept suspicious incoming friend requests, we believe that RSE attacks would become more difficult. While CAPTCHAs are not the silver bullet in preventing and stopping malicious activity on social networks (e.g., as show in [13, 23]), they do raise the difficulty bar for the attackers.

8.6 Related Work

Social engineering attacks are well-known in practice as well as in literature (e.g., [16,46,75, 93,100]). Social engineering targets human weaknesses instead of vulnerabilities in technical systems. Automated Social Engineering (ASE) is the process of automatically executing social engineering attacks. For example, spamming and phishing can be seen as a very simple form of social engineering (i.e., making users click on links).

A general problem on social networks is that it is difficult for users to judge if a friend request is trustworthy or not. Thus, users are often quick in accepting invitations from people they do not know. For example, an experiment conducted by Sophos in 2007 showed that 41% of Facebook users acknowledged a friend request from a random person [13]. More cautious users can be tricked by requests from adversaries that impersonate friends [23]. Unfortunately, once a connection is established, the attacker typically has full access to all information on the victim's profile. Moreover, users who receive messages from alleged friends are much more likely to act upon such message, for example, by clicking on links. A similar result was reported by Jagatic et al. [50]. The authors found that phishing attempts are more likely to succeed if the attacker uses stolen information from victims' friends in social networks to craft their phishing e-mails.

In contrast to active social engineering that requires the attacker to establish contact with the victim, in a reverse social engineering attack, it is the victim that contacts the attacker. We are not aware of any previous reports or studies on reverse social engineering attacks in online social networks. The results of this chapter demonstrate that automated reverse social engineering is a realistic threat, and that it is feasible in practice.

The most well-known attack to compromise the trust relationship in a social network that employs a reputation system is the *sybil attack* [33]. In this attack, the attacker creates multiple fake identities and uses them to gain a disproportionately large influence on the reputation system. Note that the findings in this chapter have implications for research that aims to defend social networks against sybil attacks (e.g., SybilGuard [106], SybilLimit [107]). SybilGuard and SybilLimit assume that real-world social networks are fast mixing [39] and this insight is used to distinguish the sybil nodes from normal nodes.

Fast mixing means that subsets of honest nodes have good connectivity to the rest of the social network. Both SybilGuard and SybilLimit are good solutions for detecting Sybil nodes. However, the attacks we present in this chapter result in legitimate friendship connections and, therefore, would not be detected by current sybil-detection approaches.

8.7 Conclusion

Hundreds of millions of users are registered to social networking sites and regularly use them features to stay in touch with friends, communicate, do online commerce, and share multimedia artifacts with other users.

To be able to make suggestions and to promote friendships, social networking sites often mine the data that has been collected about the registered users. For example, the fact that a user looks up an e-mail address might be assumed to indicate that the user knows the person who owns that e-mail account. Unfortunately, such assumptions can also be abused by attackers to influence recommendations, or to increase the chance that the victim's interest is intrigued by a fake honey-account.

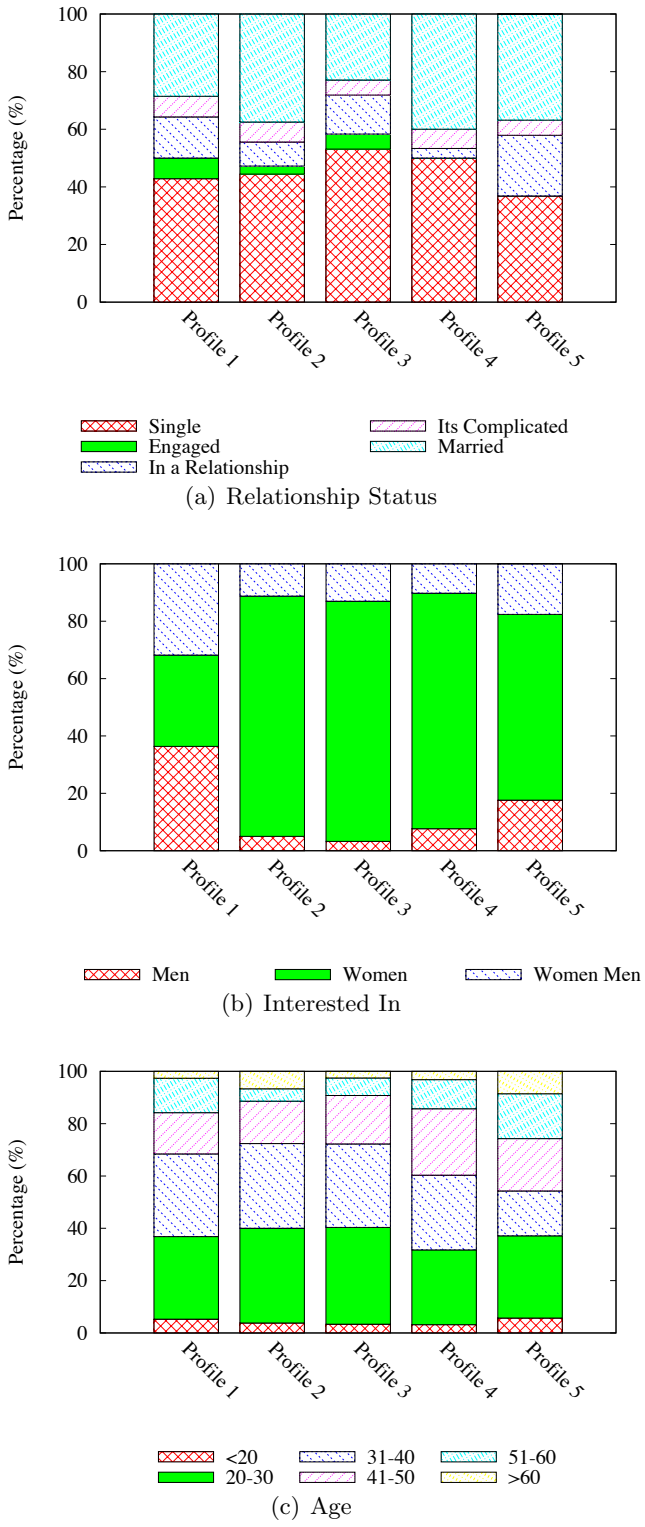
Although social engineering attacks in social networks have been well-studied to date, *reverse social engineering* (RSE) attacks have not received any attention.

This chapter presents the first user study on how attackers can abuse some of the features provided by online social networks with the aim of launching automated reverse social engineering attacks. We present and study the effectiveness and feasibility of three novel attacks: Recommendation-based, visitor tracking-based, and demographic-based reverse social engineering.

Our results show that RSE attacks are a feasible threat in real-life, and that attackers may be able to attract a large numbers of legitimate users *without* actively sending any friend request. The experiments we have conducted demonstrate that suggestions and friend-finding features (e.g., demographic-based searches) made by social networking sites may provide an incentive for the victims to contact a user if the right setting is created (e.g., an attractive photograph, an attack profile with similar interests, etc.).

We hope that this chapter will raise awareness about the real-world threat of reverse

social engineering in social networks and will encourage social network providers to adopt some countermeasures.

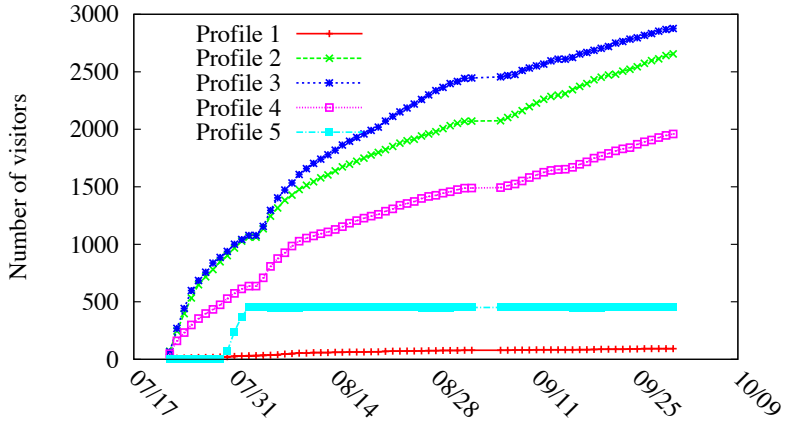


(a) Relationship Status

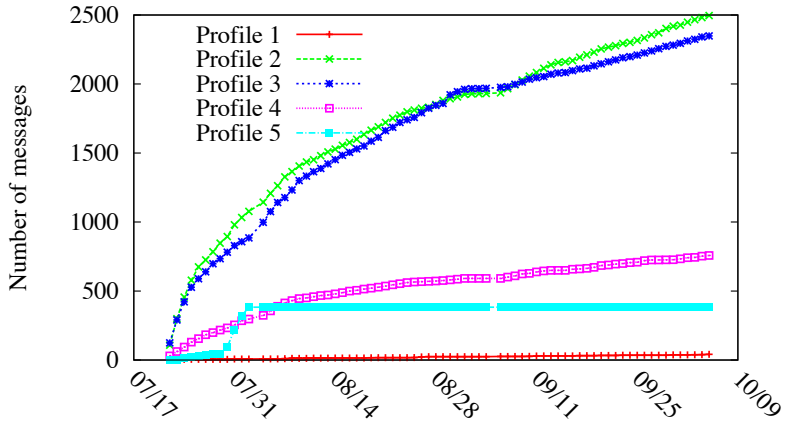
(b) Interested In

(c) Age

Figure 41: Demographic breakdown by Relationship Status, Interested In, and Age for Friend Connect requests on Facebook.

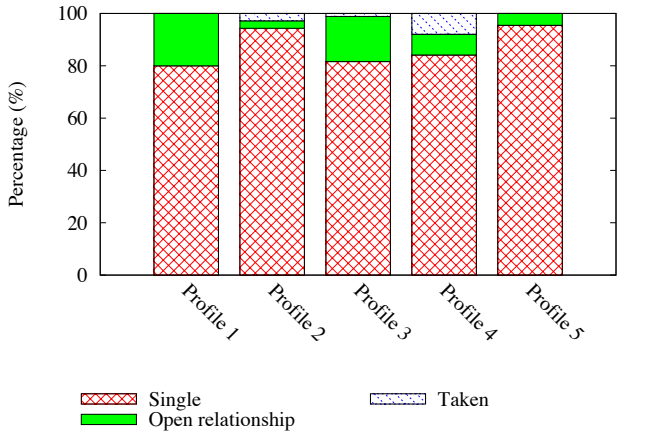


(a) Visitors to each profile

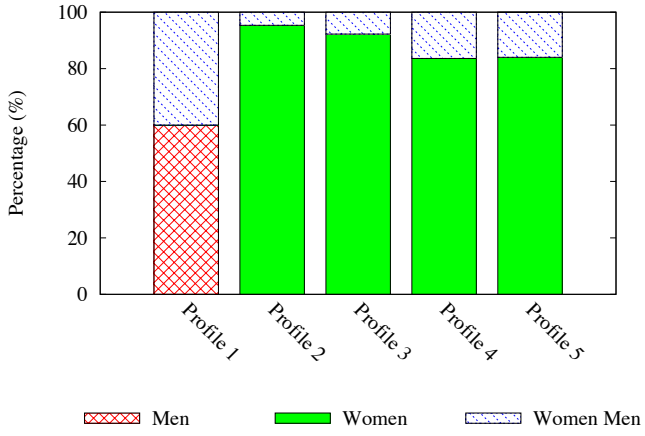


(b) Messages sent to each profile

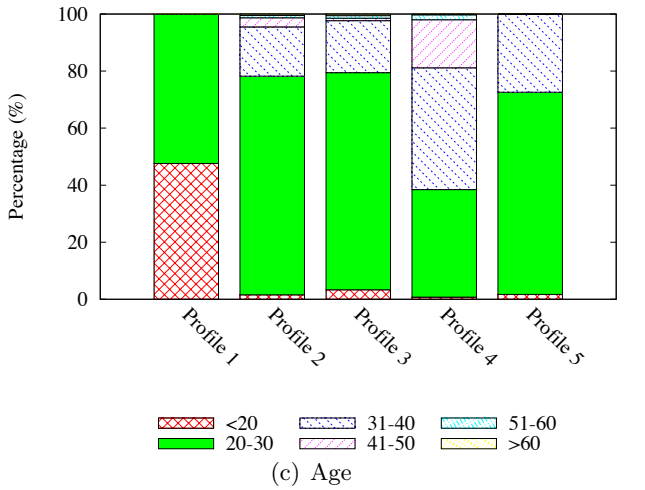
Figure 42: Cumulative counts of interactions resulting from reverse social engineering on Badoo.



(a) Relationship Status

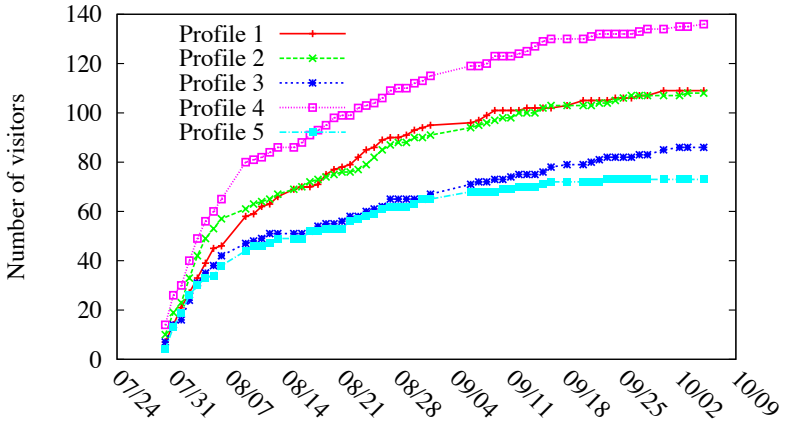


(b) Interested In

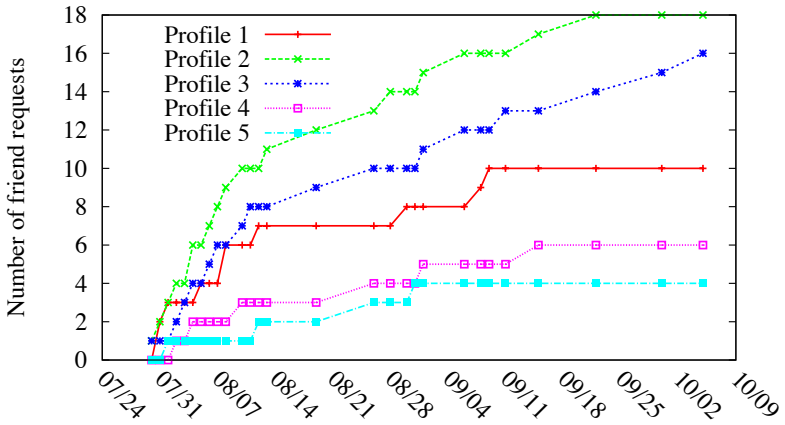


(c) Age

Figure 43: Demographic breakdown by Relationship Status, Interested In, and Age for messages on Badoo.

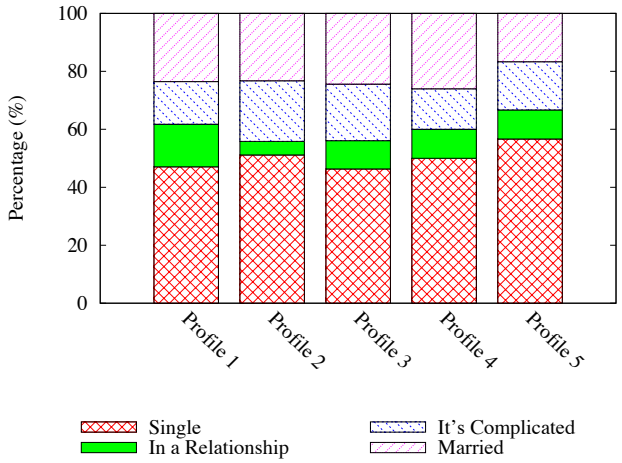


(a) Visitors to each profile

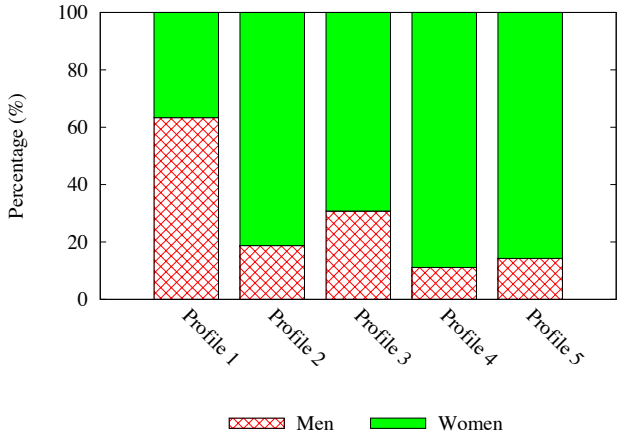


(b) Friend requests sent to each profile

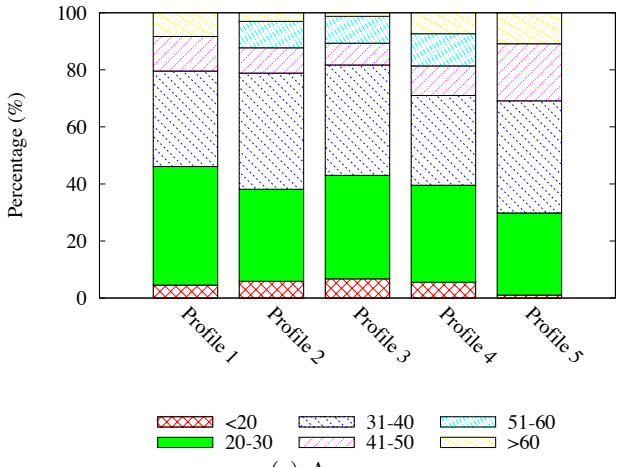
Figure 44: Cumulative counts of interactions resulting from reverse social engineering on Friendster.



(a) Relationship Status



(b) Interested In



(c) Age

CHAPTER IX

CONCLUSION

In this dissertation I have demonstrated viable approaches for preventing abuse of online communities. More specifically, I have investigated preventing against two attacks which are linked by their opposing views on low-quality information – namely, DoI attacks in which users perceive low-quality information as bad, and information leakage attacks in which users perceive low-quality information released to attackers as good.

Part I: Protection against denial of information (DoI) attacks

Low-quality information from an information consumer's standpoint

Due to the financial incentives associated with DoI attacks, adversarial adaption is an important aspect of protecting against them. We analyze phishing messages over 15 months and identify both flash and non-flash attacks based on content similarity. By looking at the construction techniques used in messages, we identified: *transitory features* that are associated with a few clusters and are short lived; and *pervasive features* that are associated with a large number of clusters and have a long life span. We find that most of the strong indicators of phishing messages turned out to be transitory. We use this to insight to identify the need for measuring classification performance with some of the strongest features removed, for any solution we propose.

To protect users from DoI attacks, we explore the effectiveness and long-term suitability of profile classification using only static content, with the aim of precluding spam content from online communities. Our results show that such a technique is definitely feasible with a standard C4.5 decision tree performs the best with an AUC of 0.99 and 0.06% misclassifications. Although the performance does decrease noticeably with the removal of the strongest features, we believe that this method is still capable of being deployed to automatically detect most spam profiles with a high confidence and mark others as gray profiles depending on the confidence of the classifier.

We also investigate protecting users against DoI attacks when content-rich profiles or content-rich messages are not available for classification (such as in the case of Twitter and classifying Tweets). We use "meta-model" classification which involves first modeling the short tweet-text (of 140 characters or less) with a meta-model of webpages linked from the tweet. The combined model increases performance in classifying tweets that belong to trends. C4.5 decision trees achieve the highest F1-measure on both the classification of tweet text and associated webpage content, resulting in an F1-measure of 0.79 and 0.9 respectively. Combining the results using an 'OR' operator provides the best results. We find that fetching associated content improves our classifiers F1-measure by over 13.7%.

Part II: Protecting against information leakage attacks

Low-quality information from a private information producer's standpoint

We approach the problem of protect users against information leakage attacks by using information unification to bring together profiles associated with users across social network sites, followed by modeling the amount of information leaked. After introducing information cloaking, we quantify the reduction in users' susceptibility towards information leakage attacks.

We introduce a technique to unify a person's profile across online communities utilizing pseudonyms. We find that over 40% of a person's online identity can be reconstructed by using a single pseudonym, and by using the person's name the attacker can reconstruct 10% to 35% of a person's online identity.

To quantify significant threat of information leakage, we introduce models to measure the amount of public information released across unified user's profiles. We find, based on over 8,200 user's online identities, that the number of users susceptible to two real-world attacks increases approximately nine times when looking at users' unified social profile in comparison to users' individual social profile. We propose four measures to quantify information revealed by a user's unified social profile. The first measure quantifies the amount of information revealed by the unified social profile; the second, quantifies the amount of information revealed and consistent in a unified social profile using Entropy from

Information Theory; the third, quantifies the amount of information revealed and consistent in a unified social profile using Guessing Entropy; and the last, quantifies the amount of consistent information revealed in a unified social profile. We use these measures to build a framework for detecting the susceptibility of a user towards an information attack, such as a personal identification attack or a password recovery attack. Using the same dataset of 8,200 user's online identities, we find that the number of users susceptible to attacks increases approximately nine times when looking at a user's combined social footprint (from 2% to 18%). We then use information cloaking to reduce the usefulness of information publicly released by reducing its granularity. For example, instead of publicly releasing an age of 25, a social network might indicate that the user is between the ages of 20-25.

Lastly, we demonstrate social engineering attacks against online communities. Our results demonstrate that reverse social engineering attacks are feasible and attackers can friend a large number of users *without* actively sending friend requests. We show that suggestions and friend-finding features (e.g., demographic-based searches) made by social networking sites may provide an incentive for the victims to contact a user if the right setting is created (e.g., an attractive photograph, an attack profile with similar interests, etc.).

REFERENCES

[1] "JunkEmailFilter.com - How it works." http://www.junkemailfilter.com/spam/how_it_works.html.

[2] "Twitter data API." http://apiwiki.twitter.com/Twitter-API-Documentation.

[3] "Twitter help – keeping search relevant." http://help.twitter.com/forums/10713/entries/42646.

[4] "Twitter help – Twitter rules." http://help.twitter.com/forums/26257/entries/18311.

[5] "Twitter: Now more than 1 billion tweets per month." http://royal.pingdom.com/2010/02/10/twitter-now-more-than-1-billion-tweets-per-month/.

[6] "Phishing: Earliest citation." http://www.wordspy.com/words/phishing.asp, Aug. 2003.

[7] "Phishtank." http://www.phishtank.com/, 2005.

[8] "APWG - Phishing activity trends: Report for December 2007." http://www.antiphishing.org/reports/apwg_report_dec_2007.pdf, 2007.

[9] "Marshal - trace report December 2007." http://www.marshal.com/newsimages/trace/Marshal_Trace_Report-Dec_2007.pdf, 2007.

[10] "Marshal - trace report July 2007." http://www.marshal.com/newsimages/trace/Marshal_Trace_Report-July_2007.pdf, July 2007.

[11] "Phishing special report: What to expect for 2007." http://www.antiphishing.org/sponsors_technical_papers/rsaPHISH2_WP_0107.pdf, Jan. 2007.

[12] "Google safe browsing API." http://code.google.com/apis/safebrowsing/, 2008.

[13] "Sophos Facebook ID Probe." http://www.sophos.com/pressoffice/news/articles/2007/08/facebook.html, 2008.

[14] "Facebook - press room: Statistics." http://www.facebook.com/press/info.php?statistics, 2009.

[15] "Facebook Statistics." http://www.facebook.com/press/info.php?statistics, 2010.

[16] "Sophos Security Threat 2010." http://www.sophos.com/sophos/docs/eng/papers/sophos-security-threat-report-jan-2010-wpna.pdf, 2010.

[17] Androutsopoulos, I., Koutsias, J., Chandrinos, K. V., Paliouras, G., and Spyropoulos, C. D., "An evaluation of naive bayesian anti-spam filtering," 2000.

[18] Balduzzi, M., Platzer, C., Holz, T., Kirda, E., Balzarotti, D., and Kruegel, C., "Abusing Social Networks for Automated User Profiling," in *Proceedings of the 13th International Symposium on Recent Advances in Intrusion Detection (RAID)*, 2010.

[19] Bayes, M. and Price, M., "An essay towards solving a problem in the doctrine of chances," *Philosophical Transactions (1683-1775)*, pp. 370–418, 1763.

[20] Beardsley, T., "Evolution of phishing attacks." `http://www.antiphishing.org/EvolutionofPhishingAttacks.pdf`, Jan. 2005.

[21] Bellman, R., "Adaptive control processes: a guided tour. 1961."

[22] Ben-Naim, A., *Entropy demystified: the second law reduced to plain common sense*, pp. 51–60. World Scientific Pub Co Inc, 2008.

[23] Bilge, L., Strufe, T., Balzarotti, D., and Kirda, E., "All Your Contacts Are Belong to Us: Automated Identity Theft Attacks on Social Networks," in *18th International Conference on World Wide Web (WWW)*, 2009.

[24] Broder, A. Z., Glassman, S. C., Manasse, M. S., and Zweig, G., "Syntactic clustering of the web," in *Proceedings of the Sixth International World Wide Web Conference*, (Santa Clara, CA), Apr. 1997.

[25] Caverlee, J., Liu, L., and Webb, S., "Socialtrust: tamper-resilient trust establishment in online communities," in *8th ACM/IEEE-CS joint conference on Digital libraries*, pp. 104–114, ACM, 2008.

[26] Caverlee, J. and Webb, S., "A large-scale study of MySpace: Observations and implications for online social networks," *International Conference on Weblogs and Social Media (ICWSM)*, vol. 8, 2008.

[27] Chew, M., Balfanz, D., and Laurie, B., "(Under)Mining Privacy in Social Networks," in *IEEE Symposium on Security and Privacy*, (Oakland, California, USA), 2008.

[28] comScore, "Social networking goes global." `http://www.comscore.com/press/release.asp?press=1555`, 2007.

[29] Cormack, G. and Lynam, T., "A study of supervised spam detection applied to eight months of personal email." `http://plg.uwaterloo.ca/~gvcormac/spamcormack.html`, 2004.

[30] Dalvi, N., Domingos, P., Sanghai, S., and Verma, D., "Adversarial classification," in *Proceedings of the 2004 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2004.

[31] Dawn, J., "Twitter blog – Get to the point: Twitter trends." `http://blog.twitter.com/2009/11/get-to-point-twitter-trends.html`.

[32] Dempster, A., Laird, N., and Rubin, D., "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 1–38, 1977.

[33] Douceur, J. R., "The Sybil Attack," in *Electronic Proceedings for the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02)*, March 2002.

[34] Farnham, S. and Churchill, E., "Faceted identity, faceted lives: social and technical issues with being yourself online," in *Proceedings of the ACM 2011 conference on Computer supported cooperative work*, pp. 359–368, ACM, 2011.

[35] Fette, I., Sadeh, N., and Tomasic, A., "Learning to detect phishing emails," in *Proceedings of the 16th International Conference on World Wide Web*, (Banff, Alberta, Canada), May 2007.

[36] Fetterly, D., Manasse, M., and Najork, M., "On the evolution of clusters of near-duplicate web pages," in *1st Latin American Web Congress*, pp. 37–45, 2003.

[37] Fetterly, D., Manasse, M., and Najork, M., "Detecting phrase-level duplication on the world wide web," in *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, (Salvador, Brazil), Aug. 2005.

[38] Fetterly, D., Manasse, M., Najork, M., and Wiener, J., "A large-scale study of the evolution of web pages," in *Proceedings of the 12th International World Wide Web Conference*, (Budapest, Hungary), May 2003.

[39] Flaxman, A., "Expansion and lack thereof in randomly perturbed graphs," *Internet Mathematics*, vol. 4, no. 2, pp. 131–147, 2007.

[40] Giles, M., "A world of connections." `http://www.economist.com/node/15351002`, Jan. 2010.

[41] Gross, R., Acquisti, A., and Heinz, III, H. J., "Information revelation and privacy in online social networks," in *WPES '05: Proceedings of the 2005 ACM workshop on Privacy in the electronic society*, (New York, NY, USA), pp. 71–80, ACM, 2005.

[42] Gyongyi, Z., Garcia-Molina, H., and Pedersen, J., "Combating web spam with trustrank," in *Thirtieth International Conference on Very large data bases (VLDB)*, pp. 576–587, 2004.

[43] Harvey, D., "Twitter blog – Trust and safety." `http://blog.twitter.com/2010/03/trust-and-safety.html`.

[44] Heymann, P., Koutrika, G., and Garcia-Molina, H., "Fighting spam on social web sites: A survey of approaches and future challenges," *Internet Computing, IEEE*, vol. 11, no. 6, pp. 36–45, 2007.

[45] Huberman, B., Romero, D., and Wu, F., "Social networks that matter: Twitter under the microscope," *First Monday*, vol. 14, no. 1-5, 2009.

[46] Irani, D., Webb, S., Giffin, J., and Pu, C., "Evolutionary study of phishing," in *eCrime Researchers Summit, 2008*, pp. 1–10, IEEE, 2008.

[47] Irani, D., Webb, S., Pu, C., and Li, K., "Study of Trend-Stuffing on Twitter through Text Classification," in *Collaboration, Electronic messaging, Anti-Abuse and Spam Conference (CEAS)*, 2010.

[48] Irani, D., Webb, S., Li, K., and Pu, C., "Large Online Social Footprints–An Emerging Threat," *Computational Science and Engineering, IEEE International Conference on*, vol. 3, pp. 271–276, 2009.

[49] Irani, D., Webb, S., and Pu, C., "Study of static classification of social spam profiles in MySpace," in *International Conference on Weblogs & Social Media*, 2010.

[50] Jagatic, T. N., Johnson, N. A., Jakobsson, M., and Menczer, F., "Social phishing," *Commun. ACM*, vol. 50, no. 10, pp. 94–100, 2007.

[51] Jakobsson, M., Finn, P., and Johnson, N., "Why and How to Perform Fraud Experiments," *Security & Privacy, IEEE*, vol. 6, pp. 66–68, March-April 2008.

[52] Jakobsson, M. and Ratkiewicz, J., "Designing ethical phishing experiments: a study of (ROT13) rOnl query features," in *15th International Conference on World Wide Web (WWW)*, 2006.

[53] Java, A., Song, X., Finin, T., and Tseng, B., "Why we twitter: understanding microblogging usage and communities," in *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis*, (New York, NY, USA), pp. 56–65, ACM, 2007.

[54] Johnson III, C., "Protection of Sensitive Agency Information," 2006.

[55] Jones, H. and Soltren, J., "Facebook: Threats to privacy," *Project MAC: MIT Project on Mathematics and Computing*, 2005.

[56] Just, M., "Designing and evaluating challenge-question systems," *IEEE Security & Privacy*, pp. 32–39, 2004.

[57] Kohavi, R., "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *International joint Conference on artificial intelligence*, vol. 14, pp. 1137–1145, 1995.

[58] Kotsiantis, S., Zaharakis, I., and Pintelas, P., "Supervised machine learning: A review of classification techniques," *Frontiers in Artificial Intelligence and Applications*, vol. 160, p. 3, 2007.

[59] Krishnamurthy, B., Gill, P., and Arlitt, M., "A few chirps about twitter," in *Proceedings of the first workshop on Online social networks*, pp. 19–24, ACM, 2008.

[60] Krishnamurthy, B. and Wills, C. E., "Characterizing privacy in online social networks," in *WOSP '08: Proceedings of the first workshop on Online social networks*, (New York, NY, USA), ACM, 2008.

[61] Kwak, H., Lee, C., Park, H., and Moon, S., "What is Twitter, a social network or a news media?," in *Proceedings of the 19th International Conference on World Wide Web*, 2010.

[62] Lauinger, T., Pankakoski, V., Balzarotti, D., and Kirda, E., "Honeybot, your man in the middle for automated social engineering," in *LEET'10, 3rd USENIX Workshop on Large-Scale Exploits and Emergent Threats, San Jose*, 2010.

[63] Lauterbach, D., Truong, H., Shah, T., and Adamic, L., "Surfing a web of trust: Reputation and reciprocity on couchsurfing.com," *Computational Science and Engineering, IEEE International Conference on*, vol. 4, pp. 346–353, 2009.

[64] Leiba, B. and Borenstein, N., "A multifaceted approach to spam reduction," in *Proceedings of the 1st Conference on Email and Anti-Spam*, (Mountain View, CA), July 2004.

[65] Lenhart, A., "Adults and social network websites." `http://goo.gl/orZF`, 2009.

[66] Li, N., Li, T., and Venkatasubramanian, S., "t-Closeness: Privacy Beyond k-Anonymity and l-diversity," *ICDE*, pp. 106–115, 2007.

[67] Machanavajjhala, A., Kifer, D., Gehrke, J., and Venkitasubramaniam, M., "l-diversity: Privacy beyond k-anonymity," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 1, no. 1, p. 3, 2007.

[68] MacQueen, J. and others, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, p. 14, California, USA, 1967.

[69] Markines, B., Cattuto, C., and Menczer, F., "Social spam detection," in *Proceedings of the Fifth International Workshop on Adversarial Information Retrieval on the Web (AIRWeb 2009)*, 2009.

[70] McCal, T., "Gartner survey shows phishing attacks escalated in 2007." `http://www.gartner.com/it/page.jsp?id=565125`, 2007.

[71] McCallum, A. and Nigam, K., "A comparison of event models for naive bayes text classification," in *AAAI-98 workshop on learning for text categorization*, vol. 752, pp. 41–48, 1998.

[72] McGiboney, M., "Twitter's tweet smell of success." `http://blog.nielsen.com/nielsenwire/online_mobile/twitters-tweet-smell-of-success/`.

[73] Michaels, S., "Facebook challenges myspace's music monopoly." `http://www.guardian.co.uk/music/2008/feb/29/news1`, 2008.

[74] Mislove, A., Marcon, M., Gummadi, K., Druschel, P., and Bhattacharjee, B., "Measurement and analysis of online social networks," in *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, p. 42, ACM, 2007.

[75] Mitnick, K., Simon, W. L., and Wozniak, S., *The Art of Deception: Controlling the Human Element of Security*. Wiley, 2002.

[76] Narayanan, A. and Shmatikov, V., "Robust de-anonymization of large sparse datasets," in *IEEE Symposium on Security and Privacy*, 2008.

[77] Narayanan, A. and Shmatikov, V., "De-anonymizing social networks," *Imprint*, 2009.

[78] Noll, M., Yeung, C., Gibbins, N., Meinel, C., and Shadbolt, N., "Telling experts from spammers: expertise ranking in folksonomies," in *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pp. 612–619, 2009.

[79] Page, L., Brin, S., Motwani, R., and Winograd, T., "The pagerank citation ranking: Bringing order to the web," tech. rep., Stanford Digital Library Technologies Project, 1998.

[80] Porter, M., "An algorithm for suffix stripping," *Program*, vol. 14, no. 3, pp. 130–137, 1980.

[81] Pu, C. and Webb, S., "Observed trends in spam construction techniques: A case study of spam evolution," in *Proceedings of the 3rd Conference on Email and Anti-Spam*, (Mountain View, CA), July 2006.

[82] Quinlan, J., *C4. 5: programs for machine learning*. Morgan kaufmann, 1993.

[83] Rabin, M., "Fingerprinting by random polynomials," tech. rep., Center for Research in Computing Technology, Harvard University, 1981.

[84] Rabkin, A., "Personal knowledge questions for fallback authentication: Security questions in the era of Facebook," in *Proceedings of the 4th symposium on Usable privacy and security*, pp. 13–23, ACM New York, NY, USA, 2008.

[85] Schechter, S., Brush, A., and Egelman, S., "It's No Secret. Measuring the Security and Reliability of Authentication via Secret Questions," in *Proceedings of the 2009 30th IEEE Symposium on Security and Privacy*, IEEE Computer Society, 2009.

[86] Sebastiani, F., "Machine learning in automated text categorization," *ACM Computing Survey*, vol. 34, no. 1, pp. 1–47, 2002.

[87] Shannon, C., "A mathematical theory of communication," *Bell Syst. Tech. J*, vol. 27, pp. 379–423, 1948.

[88] Sophos, "Facebook members bare all on networks, sophos warns of new privacy concerns." `http://goo.gl/zuLb`, 2007.

[89] Stone, B., "Twitter blog – An ongoing battle." `http://blog.twitter.com/2008/07/ongoing-battle.html`.

[90] Stone, B., "Twitter blog – Making progress on spam." `http://blog.twitter.com/2008/08/making-progress-on-spam.html`.

[91] Stone, B., "Twitter blog – What's happening." `http://blog.twitter.com/2009/11/whats-happening.html`.

[92] Stone, M., "Cross-Validatory choice and assessment of statistical predictions," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 36, no. 2, pp. 111–147, 1974.

[93] Stringhini, G., Kruegel, C., and Vigna, G., "Detecting Spammers on Social Networks," in *Annual Computer Security Applications Conference (ACSAC)*, 2010.

[94] Stringhini, G., "Spamdetector." http://www.cs.ucsb.edu/~gianluca/spamdetector.html.

[95] Sweeney, L., "Uniqueness of simple demographics in the us population," *LIDAP-WP4. Carnegie Mellon University, Laboratory for International Data Privacy, Pittsburgh, PA*, 2000.

[96] Sweeney, L., "k-anonymity: A model for protecting privacy," *International Journal of Uncertainty Fuzziness and Knowledge Based Systems*, vol. 10, no. 5, pp. 557–570, 2002.

[97] Ward, M., "Criminals exploit net phone calls." http://news.bbc.co.uk/1/hi/technology/5187518.stm, July 2006.

[98] Webb, S., Caverlee, J., and Pu, C., "Introducing the Webb Spam Corpus: Using email spam to identify web spam automatically," in *Proceedings of the Third Conference on Email and Anti-Spam (CEAS 2006)*, 2006.

[99] Webb, S., Caverlee, J., and Pu, C., "Characterizing web spam using content and http session analysis," in *Proceedings of the 4th Conference on Email and Anti-Spam*, (Mountain View, CA), Aug. 2007.

[100] Webb, S., Caverlee, J., and Pu, C., "Social honeypots: Making friends with a spammer near you," in *Fifth Conference on Email and Anti-Spam (CEAS)*, 2008.

[101] Webb, S., Caverlee, J., and Pu, C., "Granular computing system vulnerabilities: Exploring the dark side of social networking communities," in *Encyclopedia of Complexity and Systems Science*, pp. 4367–4378, Springer, 2009.

[102] Weil, K., "Twitter blog – Measuring tweets." http://blog.twitter.com/2010/02/measuring-tweets.html.

[103] Witten, I., Frank, E., Trigg, L., Hall, M., Holmes, G., and Cunningham, S., "Weka: Practical machine learning tools and techniques with Java implementations," in *ICONIP/ANZIIS/ANNES*, vol. 99, pp. 192–196, 1999.

[104] Yang, Y. and Pedersen, J., "A comparative study on feature selection in text categorization," in *Proceedings of the Fourteenth International Conference on Machine Learning*, pp. 412–420, Morgan Kaufmann Publishers Inc. San Francisco, CA, USA, 1997.

[105] Yardi, S., Romero, D., Schoenebeck, G., and boyd, d., "Detecting spam in a twitter network," *First Monday*, vol. 15, no. 1, 2010.

[106] Yu, H., Kaminsky, M., Gibbons, P., and Flaxman, A., "Sybilguard: defending against sybil attacks via social networks," in *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, pp. 267–278, ACM, 2006.

[107] Yu, H., Kaminsky, M., Gibbons, P. B., and Flaxman, A., "SybilLimit: A Near-Optimal Social Network Defense against Sybil Attacks," in *IEEE Symposium on Security and Privacy*, 2008.

[108] Zinman, A. and Donath, J., "Is Britney Spears spam?," in *Proceedings of the Fourth Conference on Email and Anti-Spam (CEAS 2007)*, 2007.